List of Acronyms

ACL	Access Control List
ANSI	American National Standards Institute
API	Application Program Interface
ASCII	American Standard Code for Information Interchange
ASW	Anti-Submarine Warfare
ATO	Air Tasking Orders
AWIS	Army WWMCCS Information System
BSD	Battlefield Situation Display
C3I	Command, Control, Communications and Intelligence
C4I	Command, Control, Communications, Computers, and Intelligence
C4IFTW	C4I For the Warrior
C4ISR	C4I Surveillance and Reconnaissance
CASE	Computer-Aided Systems Engineering
CCB	Configuration Control Board
CDE	Common Desktop Environment
CDFS	CDROM File System
CDROM	Compact Disc Read Only Memory
CDS	Cell Directory Service
CGI	Common Gateway Interface
CHS-2	U.S. Army Common Hardware and Software-2
CINC	Commander-in-Chief
CINCHQ	Commander-in-Chief Headquarters
CINFO	COE Information Server
CJTF	Command, Joint Task Force
CM	Configuration Management
COE	Common Operating Environment
COM	Common Object Management (Microsoft)
CONOPS	Concept of Operations
CONUS	Continental United States
CORBA	Common Object Request Broker Agent
COTS	Commercial Off-The-Shelf Software
CPU	Central Processing Unit
CS	Common Software
CSCI	Computer Software Configuration Item
DAT	Digital Audio Tape
DBA	Database Administrator
DBMS	Database Management System
DBO	Database Owner
DCE	Distributed Computing Environment
DCOM	Distributed Common Object Management (Microsoft)
DDDS	Defense Data Dictionary System

DDL	Data Definition Language
DDM	DOD Data Model
DEC	Digital Equipment Corporation
DES	Data Encryption Standard
DFS	Distributed File Service
DIA	Defense Intelligence Agency
DII	Defense Information Infrastructure
DISA	Defense Information Systems Agency
DISN	Defense Information Systems Network
DLC	Data Link Control
DLL	Dynamic Link Library
DMA	Defense Mapping Agency (now called NIMA)
DMS	Defense Message System
DNS	Domain Name Service
DOD	Department of Defense
DTS	Distributed Time Service
EC/EDI	Electronic Commerce/Electronic Data Interchange
ECP	Engineering Change Proposal
ECPN	Electronic Commerce Processing Node
EPAC	Extended Privilege Attribute Certificate
ESL	Event Selection List
EUCOM	US European Command
FTP	File Transfer Protocol
GCCS	Global Command and Control System
GCSS	Global Combat Support System
GDI	Graphics Display Interface
GDS	Global Directory Service
GEOLOC	Geographical Location File
GIF	Graphics Interchange Format
GOTS	Government Off-The-Shelf Software
GPS	Global Positioning Satellite
GSC	GCCS Site Coordinator
GSORTS	GCCS Status of Resources and Training System
GSSAPI	Generic Security Services API
GUI	Graphical User Interface
HP	Hewlett Packard
HPFS	High Performance File System
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
I&RTS	Integration and Runtime Specification
IBM	International Business Machine
IDB	Integrated Database

IDL	Interface Definition Language
IP	Internet Protocol
IS	Information System
JCALS	Joint Computer-Aided Acquisition and Logistics Support
JIEO	Joint Interoperability and Engineering Organization
JMCIS	Joint Maritime Command Information System
JOPES	Joint Operations Planning and Execution System
JPEG	Joint Photographic Expert Group
JSS	Joint Shared Servers
JTA	Joint Technical Architecture
JTF	Joint Task Force
JWID	Joint Warrior Interoperability Demonstration
KG	Keying Generator
KPC	Kernel Platform Certification
LAN	Local Area Network
MB	Megabyte
MCG&I	Mapping, Charting, Geodesy, and Imaging
MSN	Microsoft Network
MUX	Multiplexer
NCA	National Command Authority
NDI	Non-Developmental Item
NES	Network Encryption Standard
NFS	Network File Server
NIMA	National Imagery and Mapping Agency (formerly called DMA)
NTDS	Naval Tactical Data System
NTFS	NT File System
ODBC	Open Database Connectivity
OLE	Object Linking and Embedding
OPLAN	Operations Plan
OS	Operating System
OSD	Office of the Secretary of Defense
OSF	Open Software Foundation; Operational Support Facility
OTH-GOLD	Over-the-Horizon GOLD
PAC	Privilege Attribute Certificate
PACAF	Pacific Air Forces
PACOM	US Pacific Command
PC	Personal Computer
POSIX	Portable Operating System for Information Exchange

RAM	Random Access Memory
RDBMS	Relational Database Management System
RM	Reference Monitor
RPC	Remote Procedure Call
RTE	Runtime Environment
S&M	Scheduling and Movement
SDMS	Software Distribution Management System
SDS	Shared Data Server
SHADE	Shared Data Environment
SIPRNET	Secret Internet Protocol Router Network
SOP	Standard Operating Procedure
SQL	Structured Query Language
SSA	Software Support Activity
SVGA	Super Video Graphics Adapter
TAFIM	Technical Architecture Framework for Information Management
TBMCS	Theater Battle Management Control System
TCP	Transmission Control Protocol
TDA	Tactical Decision Aid
TPFDD	Time-Phased Force and Deployment Data
TRANSCOM	US Transportation Command
TRM	Technical Reference Manual
UDP	User Datagram Protocol
UIS	User Interface Specification
URL	Uniform Resource Locator
USMTF	United States Message Text Format
UTM	Universal Transverse Mercator
UUID	Unique Universal Identifier
VFAT	Long Filename File Allocation Table
VGA	Video Graphics Adapter
VRML	Virtual Reality Modeling Language
WAN	Wide Area Network
WWMCCS	World-Wide Military Command and Control System
WWW	World-Wide-Web
WYSIWYG	What-You-See-Is-What-You-Get

Appendix A: Supported Configurations

The COE is an open architecture and as such is not tied to a specific hardware platform. It emphasizes use of industry standard software components, including a POSIX-compliant operating system and windowing standards. However, availability of sufficient resources to perform adequate testing necessitates a practical limit on the number of configurations supported.

The list of supported hardware and software components is growing as the COE evolves to meet operational requirements. This appendix will be updated periodically as new platforms are supported and as vendor software versions change. An up-to-date list of all COTS products and their version numbers is included with each COE release and with each release of a COE-based system. **Refer to the most recent *DII COE Baseline Specifications* document for an up-to-date description of supported platforms and COTS product versions.**

Platforms Supported

This section describes the current configurations for UNIX and NT platforms.

UNIX Configurations

The COE is available on Hewlett-Packard HP 700, 800, and 9000 series computers, and on Sun SPARC series computers. Work to port the COE to other UNIX platforms is underway, but at the time of the writing of this document, they have not been released. Additional platforms include IBM RISC 6000, DEC Alpha, and SGI computers. The Kernel Platform Certification program described in this appendix provides an avenue for presently unsupported platforms. Contact the DII COE Chief Engineer for availability of other platforms.

Table A-1 lists the present configurations for DII COE 3.0 and for selected COTS products contained in the COE. The presently selected vendor is indicated for COTS products. This list is not intended to be exhaustive. It lists only those products that are part of the COE kernel, or are part of Infrastructure Services.

Precise hardware requirements for memory, disk space, etc. is a function of whether the platform is a database server, an application server, or client platform, and whether the platform is standalone or on a network. Refer to the DII COE Chief Engineer for hardware configuration options.

	HP Series	**SPARC Series**
Operating System	HP-UX 9.0.7, 10.10	Solaris 2.4, 2.5.1
X Windows	X11R5	X11R5
Motif[1]	1.2.4	1.2.4
CDE	4.2 (HP)	4.2 (TriTeal)
Transarc DCE	1.1	1.1
NewsPrint	N/A	2.5
Sybase	10.0.2a	10.0.2a
Oracle	7.2.3.2	7.2.3.2
Informix	7.2	7.2
Netscape Browser	3.0	3.0

Table A-1: Supported UNIX Configurations

[1] Developers should use the Motif libraries distributed with CDE.

Note: An upgrade to HP-UX 10.30, X11R6, and Motif 2.x are planned. Upgrades to more recent Oracle, Sybase, and Informix products are also planned, but must be coordinated with affected services and agencies. All COTS upgrades are coordinated through the DII COE CCB so that affected services/agencies are well aware of the planned upgrades so that they can participate in the scheduled release of product upgrades. Contact the DII COE Chief Engineer for target release dates.

NT Configurations

The COE for NT platforms is presently available only for Intel-based computers (e.g., 80x86, Pentium). Commercial industry has implemented the Microsoft NT operating system on selected other platforms (e.g., DEC), but such platforms are not presently in wide use in the DII community.

On PCs, only NT is supported. Windows 3.1 and Windows 3.11 are not supported. Limited testing of the COE has been performed on Windows 95, but it is not currently a supported platform because of known security problems within the operating system. When the security problems are resolved, Windows 95 may be added to the list of supported platforms.

The COE requires that all hardware be NT-compliant (as defined by the document *Microsoft Windows NT Hardware Compatibility List #4094*), because Microsoft does not guarantee that NT will run on systems with hardware not listed in the referenced document. Contact DISA for JIEO Report 8300, *Department of Defense Minimum Desktop Personal Computer Configuration*, dated 19 July 1996 for a description of minimum hardware requirements for a PC platform.

Table A-2 lists the presently supported products for the NT platform. This list is not intended to be exhaustive. It lists only those products that are part of the COE kernel or are part of Infrastructure Services.

Proper operation of the COE also requires installation of the appropriate Microsoft Service Pack. Microsoft periodically releases Service Packs (e.g., patches). Refer to the latest *DII COE Release Notes for NT* for information about the required Service Pack.

	NT (80x86, Pentium Only)
Operating System	NT 4.0
Transarc DCE	1.1
Netscape Browser	3.0

Table A-2: Supported NT Configurations

DII COE Kernel Platform Certification Program

The purpose of the DII COE Kernel Platform Certification (KPC) program is to provide criteria and a process for certification of POSIX-based application platforms as DII COE Compliant. A document which describes the program,

· *Defense Information Infrastructure (DII) Common Operating Environment (COE) Kernel Platform Certification Program*, Draft version 0.6 - June 1997,

may be found at URL

`http://spider.osfl.disa.mil/dii`.

Criteria for DII COE Kernel Platform Certification are defined in accordance with the following engineering documents:

· *Defense Information Infrastructure (DII) Common Operating Environment (COE) Integrated Run-Time Specification (I&RTS)* (this document), and

· *Defense Information Infrastructure (DII) Common Operating Environment (COE) Baseline Specifications*, (most recent version).

The *DII COE Baseline Specifications* contains a consolidated list of all software available in the DII COE, independent of any specific application platform implementation. COE kernel components are those elements required to be present on all DII COE application platforms, and are listed in the *Baseline Specifications*.

Objectives

This program will establish a process which encourages Information Technology (IT) industry suppliers to provide DII COE kernel platform functionality in their application platform products. DISA will investigate an applicant claim of conformance and make a list of application platforms with valid certificates available to the public.

An additional objective of this program relates to the Government-supplied software included in the DII kernel platform. This software provides features and functions which are essential to the operation of an application platform within a distributed system. DISA would prefer that these features were provided within commercial operating system offerings. This objective will be satisfied when commercial offerings replace the government supplied software in the DII COE kernel.

Applicability of Certification

The DII COE kernel platform certification process is available for POSIX-based DII COE application platforms. Note that this program applies only to the DII COE kernel, not to the entire DII COE.

This certification program provides an opportunity for vendors of POSIX-based application platforms to enhance the appeal of their product for DOD customers with a need for such platforms. DOD CINCs, Services, and Agencies which procure and use POSIX-based DII COE application platforms may use this certification program as one measure of the suitability of the product.

Comments or questions regarding the Kernel Platform Certification program may be directed to the point of contact identified on the final page of the *DII COE Kernel Platform Certification Program* document referenced above.

Program Restrictions

There are certain restrictions to the self-certification program, whether requested by a vendor or by a program manager.

1. The program applies only to the DII COE kernel, not to the entire COE.

2. Source code is provided only for GOTS products, not COTS products, in the kernel.

3. The porting package is provided on an "as-is" basis with no guarantees or estimates on DISA's part for the amount of effort required to do the port.

4. The requester must sign a non-disclosure, non-distribution agreement to receive the source code.

5. The requester may not extend the kernel by adding new features to the GOTS products.[2]

6. Continued support for the new environment is the responsibility of the requester. DISA, at its discretion, may elect to provide support for the new environment depending upon requirements established by the DII community.

7. The requester must submit the completed port to the DII COE SSA for certification as COE compliant.

8. The requester is responsible for obtaining any COTS products required for the kernel and for any associated licensing required while DISA tests the port for certification purposes. The COTS products obtained must be approved by the DII COE Chief Engineer as suitable functional equivalents of COTS products used in the kernel for supported platforms.

[2] The restriction to not provide "value-added" capabilities is an important one. The intent is to avoid vendor-unique implementations which introduce capabilities in one environment that are not available on another. Such a situation would defeat the whole purpose of having a COE in the first place.

This page is intentionally blank.

Appendix B: Compliance Checklists

DII compliance is the cornerstone to ensure seamless segment integration and proper system operation. An objective technique for measuring DII compliance is required. Such a system has the following advantages:

· It allows quantitative statements to be made about whether or not a segment or COE-based system is compliant. It also provides an objective measure of the *degree* to which a segment or system is DII-compliant.

· It serves to identify areas in which segments need to improve to achieve compliance. A side benefit is the identification of potential areas of interoperability problems because of identifying areas where the system overlaps COE functionality.

· It provides a meaningful way to quantitatively compare segments in addition to traditional measures such as functional coverage. One important dimension to this technique is that it directly incorporates interoperability comparisons between segments being evaluated.

· It aids in developing a migration strategy for legacy systems. Program managers can use the checklist to determine where they stand with regards to compliance and then use the identified areas of non-compliance to create a strategy and schedule for achieving the target level of compliance.

This appendix addresses primarily Category 1 (Runtime Environment) compliance, although there are also certain Category 2 (Style Guide) compliance items listed. The *DII User Interface Specification* addresses Category 2 compliance and also has a checklist organized to complement the checklist given here. Refer to that document for specific

Category 2 requirements and for how they map to the equivalent Category 1 levels presented here.

Chapter 2 defines Category 1 compliance as eight levels of progressively deeper integration, because compliance cannot be an all-or-nothing proposition for legacy systems. The levels progress from a state of “peaceful coexistence” to “federation of systems” to true integration. Chapter 2 also shows how the levels of compliance map to levels of interoperability, and that interoperability increases as the level of DII compliance increases. As noted in Chapter 2, segments shall achieve Level 7 compliance before being accepted or advertised as an approved DISA product. At DISA’s discretion, segments that are Level 5 or Level 6 compliant may be accepted as prototypes and fielded at selected sites for evaluation purposes. Developers must achieve Level 4 compliance before DISA will consider evaluating a prototype for eventual migration into the COE or COE-based DISA systems.

This appendix contains a series of questions, in a checklist format, which are organized by compliance level. The philosophy behind this design for the compliance checklists is to begin with an agreement on a set of standards[3], ensure non-interference when installed on the same LAN, then non-interference when installed on the same platform, and finally to interoperability through sharing the same software and data. With this strategy, it is possible to define a minimally acceptable level of compliance that balances system risk against cost to achieve full compliance.

Segments shall be evaluated against these checklists to determine the degree of compliance. There are several things to note in order to properly apply the compliance checklists in this appendix.

1. System components are not required to be in segment format until Level 4. Therefore, the compliance checklists use the generic term “application” to refer to any system component being evaluated in levels 1-3. Beginning with Level 4, they are properly referred to as segments because compliance at Level 4 requires segmentation.

2. Each item in the checklists shall be answered as *True*, *False*, or *Not Applicable* as appropriate. Not all questions are necessarily applicable to all segments, and hence should be marked as “N/A.” For example, questions that deal with database issues are not applicable to applications/segments that do not use a database. Similarly, questions that deal with Motif are not applicable to applications/segments that run on NT platforms, or which do not provide a GUI.

3. The Category 1 compliance level assigned to the application/segment is the highest numbered level for which there are no “False” replies. As explained in Chapter 2, it is not permissible to describe an application/segment in a manner such as “80% Level 6” compliant.

[3] The *JTA* contains a standards profile that compliant systems must adhere to. However, to allow for migration of legacy systems, full compliance is not stipulated in Level 1 but is spread judiciously across all 8 levels.

4. Note that each question is prefaced by the level it applies to. Thus, question 6-3 refers to the third compliance question for Level 6.

5. Each question in the checklists is independent of the operating environment (e.g., UNIX versus NT) unless noted as environment-specific. Environment-specific questions are noted by including "(UNIX)" or "(NT)" as appropriate at the beginning of the question.

6. Each successive level is inclusive. For example, Level 6 compliance means that the segment has achieved compliance for levels 1-5.

7. In the checklists there are references made to approvals by the Chief Engineer. Unless otherwise qualified, this means the DII COE Chief Engineer for COE-component segments and the cognizant DOD program Chief Engineer for mission-application segments. The principle being applied is that any modification that affects interoperability or the COE requires approval by the DII COE Chief Engineer. All other approvals are the domain of the program Chief Engineer because they are limited in scope to the system being built.

8. In the checklists there are references to the SSA. Unless otherwise qualified, this means the DII COE SSA for COE-component segments and the cognizant DOD program SSA for mission-application segments.

9. In each case, it is the intent that "sanity is to prevail." Applications/segments are expected to comply with the requirements specified, but the DII COE Chief Engineer may grant unusual exceptions on a case-by-case basis.

10. The COE is presently available for NT only on Intel-based platforms (e.g., 80x86, Pentium). The NT questions in the checklist are applicable only to that hardware environment and will be upgraded as required if NT support is made available for non-Intel platforms. The concept conveyed by the checklist items are applicable regardless of the hardware platform or operating system, but for clarity are often worded in such a way as to make the statement operating-environment-specific.

In general, applications should be able to reach Level 5 compliance without requiring source code changes. This is true only if good programming practices are followed. For example, use of hardcoded pathnames is not a good programming practice. It would thus likely require source code changes to achieve Level 5 compliance because of the requirement that the segment use relative filenames for files within the segment (see the *Runtime Environment* series of questions below under Level 5 compliance).

The checklists presented in this appendix are organized in a very deliberate way and with a deliberate objective in mind of evaluating DII compliance. There are several points about how the checklists are organized.

· The checklists are organized in such a way as to evaluate individual segments. However, as Chapter 2 describes, the compliance of individual segments can be combined into a composite compliance level. Many of the items in the appendix are applicable to only specific types of segments (e.g., COE-component segments, account group segments), and should be noted as “not applicable” for many systems.

· The organization presented here is not the only possible, nor the only useful, way of organizing a compliance checklist. It could be organized by segment type, but such an approach leads to duplication of checklist items among the various segment types. It could also be organized by how the segment will be used in the resulting system (e.g., user presentation, data server, application server), but this suffers the same problem as organization by segment type and moreover, the compliance criteria for a segment should be independent of how the segment will ultimately be used in the target system.

· The organization presented here does not directly address the amount of work required to reach any specific level of compliance. The reason it does not attempt to do so is because the effort required to reach compliance is heavily dependent upon the system under evaluation, and the areas in which the system is weak with respect to compliance. For example, a UNIX system that is not based on Motif will likely require a substantial amount of effort to achieve Level 7 or 8 compliance even though there are only a few checklist questions related to Motif. On the other hand, there are several specific questions related to how the system is packaged into segments, but this generally requires little or no coding changes at all to achieve compliance.

· The organization presented here is conducive to pinpointing problem areas both from a compliance perspective, and as a side effect, from an interoperability perspective. Part of the effort in achieving compliance is to ensure conformance to standards, and to ensure that there is no duplication of COE services. This directly leads to pinpointing potential interoperability problem areas.

· The COE specifically avoids weighting compliance questions. This has the obvious disadvantage of assigning equal importance to each question within a level, but the reverse problem is that assigning weights cannot be done outside the context of the objectives of the end system. For example, if the objective is to migrate a legacy system rapidly so that it can be deployed quickly, that leads to a different set of weights than if the objective is to achieve full interoperability.

Program managers may wish to organize the compliance checklists presented here in a manner more conducive to meeting programmatic objectives. However, neither the compliance levels nor the applicable questions at each level shall be modified.

Standards Compliance (Level 1)

				Standards Compliance
T	**F**	**N/A**	**1-1**	(NT) Hardware components are Windows NT-compliant as defined by the Microsoft document *Microsoft Windows NT Hardware Compatibility List #4094*.
				Operating System
T	**F**	**N/A**	**1-2**	The operating system and associated software conform to the following standards from the *JTA*: (a) ISO 9445-1:1996, Information Technology - Portable Operating System Interface for Computer Environment (POSIX) - Part 1: System Application Program Interface (API) [C Language], as profiled by FIPS 151-2:1994. (b) IEEE 1003.1g:1996 Draft, POSIX - Part 1: System Application Program Interface (API) Amendment 2: Protocol Independent Interfaces (Sockets) [C Language].
T	**F**	**N/A**	**1-3**	Unless approved by the DII COE Chief Engineer, the operating system supports the `System` API for FIPS 119 (Ada95).
T	**F**	**N/A**	**1-4**	The operating system is configured to support DCE.
T	**F**	**N/A**	**1-5**	The operating system is configured to support TCP/IP protocols.
T	**F**	**N/A**	**1-6**	The operating system is configured to support UDP protocols.
T	**F**	**N/A**	**1-7**	The operating system is configured to support SLIP and PPP.
T	**F**	**N/A**	**1-8**	Custom device drivers added to support program-unique requirements, if any, do not interfere with native capabilities of the operating system no do they cause a violation of other mandated standards for the operating system or network.
T	**F**	**N/A**	**1-9**	(NT) The operating system is the same version as provided by the COE, or higher (see Appendix A).
				Network Services
T	**F**	**N/A**	**1-10**	The application can execute in an environment that includes DII COE-provided DCE services.
T	**F**	**N/A**	**1-11**	The application uses only those TCP/IP interfaces provided by the native operating system.
T	**F**	**N/A**	**1-12**	The application uses only those UDP or point-to-point interfaces provided by the native operating system.
T	**F**	**N/A**	**1-13**	The application uses only those SLIP or PPP interfaces provided by the native operating system.

GUI Environment				
T	**F**	**N/A**	**1-14**	The application complies with the style of the native GUI. (See GUI compliance requirements in the *DII User Interface Specification*.)
T	**F**	**N/A**	**1-15**	The windowing environment conforms to the following standard from the *JTA*: ISO 9945-2: 1993, Information Technology - Portable Operating System Interface for Computer Environments (POSIX) - Part 2: Shell and Utilities as profiled by FIPS PUB 189:1994.
T	**F**	**N/A**	**1-16**	(UNIX) The windowing environment conforms to the following standard from the *JTA*: FIPS Pub 158-1:1993, User Interface Component of the Application Portability Profile X-Windows Version 11, Release 5.
Database Services				
T	**F**	**N/A**	**1-17**	If an RDBMS is used, it supports FIPS-127-2 SQL queries.

Network Compliance (Level 2)

Security				
T	**F**	**N/A**	**2-1**	The application is able to operate correctly with the operating system security modules enabled (BSM for Solaris, C2 enabled for HP, etc.).
Operating System				
T	**F**	**N/A**	**2-2**	The operating system supports NFS servers and clients.
T	**F**	**N/A**	**2-3**	(UNIX) The operating system can be configured to support DNS/NIS/NIS+. (Note: The requirement is that the operating system be capable of supporting centralized management of key resources such as hostnames, user accounts, etc. NIS+ is not a specific requirement because not all vendors support it.)
T	**F**	**N/A**	**2-4**	(NT) NT is configured to use the NTFS file system for files stored on hard disks. (Note: NT uses the FAT file system for floppy diskettes. Such usage is generally transparent to applications. However, NTFS is required on the hard disk for security reasons.)
Network Services				
T	**F**	**N/A**	**2-5**	The operating system supports sockets, including Berkeley sockets.
T	**F**	**N/A**	**2-6**	The application is able to operate properly in an environment where other applications are performing UDP broadcasts.
T	**F**	**N/A**	**2-7**	The application does not require any particular hostname conventions nor does it need reserved IP addresses.
T	**F**	**N/A**	**2-8**	The ability of the application to execute correctly is independent of the type of LAN (e.g., Class B or Class C) connected to the platform.
T	**F**	**N/A**	**2-9**	(UNIX) The application can operate in a DNS/NIS/NIS+ environment. (Note: The requirement is that the application be able to operate correctly when the features supported by the operating system for centralized management of key resources are enabled.)
T	**F**	**N/A**	**2-10**	(NT) If the target system is configured to use Microsoft domains and workgroups, the application can operate correctly in such an environment.
T	**F**	**N/A**	**2-11**	(NT) The application uses native PC byte order for data internal to the PC, but uses network byte order for data external to the PC.
T	**F**	**N/A**	**2-12**	(NT) The application uses native PC byte order to access `$DATA_DIR/local` and `$DATA_DIR/PCglobal` PC data.
T	**F**	**N/A**	**2-13**	(NT) The application uses network byte order to access `$DATA_DIR/global` data.

GUI Environment				
T	**F**	**N/A**	**2-14**	(UNIX) If the application is an X Windows application, it is compatible with the X server supplied by the COE (see Appendix A).
Database Services				
T	**F**	**N/A**	**2-15**	Database updates operate correctly with DBMS security audits enabled.
T	**F**	**N/A**	**2-16**	The database is recoverable to a consistent state in the event of DBMS server, network, or client application failure. This includes both hardware and software failures.
T	**F**	**N/A**	**2-17**	Database transactions implement strict two-phase locking .

Platform Compliance (Level 3)

Operating System		
T F N/A	**3-1**	If extensions to the operating system as configured for the COE are required, all such extensions have been identified and documented. This includes the configuration of all operating system resources including the amount of shared memory required, the number of semaphores, the message queue size, etc.
T F N/A	**3-2**	The operating system configuration required by the application does not decrease or con flict with any system resources as already configured for the COE. The application may increase system resource configurations, but not decrease them.
T F N/A	**3-3**	The application does not use hardcoded port assignments (e.g., from `/etc/services`) and is not sensitive to specific ports other than well-known port assignments (e.g., `ftp`, `listen`). If the application uses network services, including standard services such as `ftp` and `listen` as well as its own private services, it retrieves the port number(s) by service name from the `/etc/services` file.
Network Services		
T F N/A	**3-4**	If the application uses ftp, it can operate in an environment where only anonymous ftp is available.
GUI Environment		
T F N/A	**3-5**	(UNIX) The application does not make direct calls to X libraries that conflict with applications that use Motif libraries to access lower-level X functions. For example, the application does not use lower-level X library functions to establish window border style or colors that either conflict with or override settings established by Motif.
T F N/A	**3-6**	(UNIX) The application does not alter any files in the vendor-supplied X or Motif directories (e.g., modify `rgb.txt` or `Xdefaults`) unless authorized by the DII COE Chief Engineer. Approval by the DII COE Chief Engineer is required because of the potential effect of the presegmented application on other segments running on the same platform.
T F N/A	**3-7**	(UNIX) The application can use the same X server version and `xdm` version that is supplied by the COE (see Appendix A).
T F N/A	**3-8**	(UNIX) The application uses either the same version of Motif as provided by the COE (see Appendix A) or does a static link to Motif libraries so that it does not conflict with other COE-based segments.
T F N/A	**3-9**	(NT) The application uses the same version of NT as supported by the COE (see Appendix A).
T F N/A	**3-10**	(NT) Unless a COTS application, the application uses only Win32 APIs to access Windows routines.

				Database Services
T	**F**	**N/A**	**3-11**	The application does not modify the user’s DBMS environment as established by the DBMS COE-component segment.
				DCE Services
T	**F**	**N/A**	**3-12**	If using RPCs, the application is compatible with the RPC mechanisms supported by the DCE version supplied by the COE.
				COTS Products
T	**F**	**N/A**	**3-13**	The software is capable of running in an environment that includes DII COE approved COTS products as specified in the *DII COE Baseline Document* for the COE version being used.
T	**F**	**N/A**	**3-14**	Configuration changes made to COTS products, if any, do not render inoperable any features available to COE-based segments or users that are already using the COTS product.
T	**F**	**N/A**	**3-15**	The application does not require any source code modifications to COTS products, except as authorized by the DII COE Chief Engineer. (Some commercial products, such as `xdm`, may require modification for security reasons. `xdm` has already been modified by the DII COE Chief Engineer to address security concerns.)
T	**F**	**N/A**	**3-16**	(NT) If the application is a COTS product that uses 16-bit APIs, there is no 32-bit alternative.
				Runtime Environment
T	**F**	**N/A**	**3-17**	The application does not alter any files outside its own directory in such a way that it conflicts with any other COE-based segment.
T	**F**	**N/A**	**3-18**	The application can operate on a COE-configured platform without altering the location or version of any system software (UNIX, X Windows, Motif, NT, etc.).
				Miscellaneous
T	**F**	**N/A**	**3-19**	(NT) The application supports VGA and SVGA resolutions.
T	**F**	**N/A**	**3-20**	(NT) The application supports 16x16, 32x32, and 64x64 icons.

Bootstrap Compliance (Level 4)

Security				
T	**F**	**N/A**	**4-1**	If an aggregate segment, the security level of the parent dominates the security level of the children.
T	**F**	**N/A**	**4-2**	Documentation is submitted with the segment that clearly identifies releasability restrictions.
Standards Compliance				
T	**F**	**N/A**	**4-3**	All software and data are packaged in segment format.
T	**F**	**N/A**	**4-4**	The segment successfully passes `VerifySeg` with no errors. Warnings are acceptable but the reason for them must be documented in the `IntegNotes` file.
T	**F**	**N/A**	**4-5**	The segment uses the COE kernel provided by the COE, or all extensions required are documented and handled by the segment in such a way that it does not interfere with other segments. For example, community files are not destructively overwritten by the segment because other segments may also need to made alterations to the community file during their own installation.
T	**F**	**N/A**	**4-6**	The segment can be installed and removed completely through the COE installation tools. If the segment is a “permanent” segment (i.e., it has no `DEINSTALL` file. See Chapter 5) and is not a candidate for removal, the segment has been tested to ensure that upgrades successfully preserve data files that must be retained during upgrades.
T	**F**	**N/A**	**4-7**	(NT) Unless a COTS segment, the segment does not modify the root-level `AUTOEXEC.BAT`, `CONFIG.SYS`, `AUTOEXEC.NT`, or `CONFIG.NT` files.
T	**F**	**N/A**	**4-8**	(NT) Unless a COTS segment, the segment does not modify `WINDOWS.INI` and `SYSTEM.INI`. The segment may freely modify its own `.INI` files.
Database Services				
T	**F**	**N/A**	**4-9**	Database owners do not use system storage areas during database creation.
T	**F**	**N/A**	**4-10**	The segment does not modify the core database storage areas, create objects in system storage areas, or create objects in public storage areas (e.g., create rollback table space).
COTS Products				
T	**F**	**N/A**	**4-11**	The segment uses the same COTS configurations as those specified by the applicable *DII COE Baseline Document* for any COTS product it uses that may also reside on the platform.

Runtime Environment				
T	**F**	**N/A**	**4-12**	Runtime extensions to the COE required by the segment have been identified and documented.
T	**F**	**N/A**	**4-13**	The segment uses the same runtime environment configuration as provided by the COE with extensions, if any, made through environment extension files and segment descriptors.
T	**F**	**N/A**	**4-14**	The segment uses the same versions, configurations, patches, and file locations as provided by the COE for all components of the COE kernel.
T	**F**	**N/A**	**4-15**	The segment uses the DII COE directory layout or a migration plan to achieve proper directory layout has been prepared.
T	**F**	**N/A**	**4-16**	(NT) The segment is able to handle Unicode filenames.

Minimal DII Compliance (Level 5)

Security		
T F N/A	**5-1**	For COE-component segments, prior approval has been granted by the DII COE Chief Engineer to provide a command-line mode or feature. The `$CMDLINE` keyword is used in the `Direct` segment descriptor to indicate command-line access is provided.
T F N/A	**5-2**	For mission-application segments, p rior approval has been granted by the Chief Engineer to provide a command-line mode or feature. The `$CMDLINE` keyword is used in the `Direct` segment descriptor to indicate command-line access is provided.
T F N/A	**5-3**	For all segments, whether COE-component segments or mission-application segments, prior approval has been granted by the DII COE Chief Engineer to provide a command-line mode or feature that provides “superuser” access. The `$CMDLINE` and `$SUPERUSER` keywords are used in the `Direct` segment descriptor to indicate superuser access.
T F N/A	**5-4**	The segment does not provide a “back door” access to a command-line prompt. If a command-line mode is available, it is through a known, documented approach for all authorized users and not through some hidden, undocumented approach.
T F N/A	**5-5**	If privileged user permissions are required during segment installation or removal (e.g., use of the `$ROOT` keyword), prior approval has been granted by the Chief Engineer.
T F N/A	**5-6**	(UNIX) The segment does not alter the COE established `umask` setting.
Standards Compliance		
T F N/A	**5-7**	The segment uses the same COE kernel as provided by the COE and documented in the applicable *DII COE Baseline Document* for the COE version being used.
T F N/A	**5-8**	All directory and file names contain only p rintable, non-blank, standard ASCII characters.
T F N/A	**5-9**	The segment does not create user login accounts. (This does not apply to the account group segments that are part of the COE kernel, but it *does* apply to all other account group segments. This also does not restrict segments from creating “non-login accounts” for use in establishing a segment group id.)
T F N/A	**5-10**	The segment can operate in an environment where user accounts are created and deleted at any time by the site administrator responsible for managing user accounts. The segment accounts for this and creates and initializes operator preferences the first time the segment is activated after a new account is created.
T F N/A	**5-11**	The segment loads correctly into the directory assigned by the COE installation tools. It does not require being loaded in any specific directory unless the Chief Engineer has granted a waiver. (This requirement does *not* apply to COTS segments.)

T	**F**	**N/A**	**5-12**	The segment conforms to the COE version numbering scheme.
T	**F**	**N/A**	**5-13**	The segment does not move directories or files from the segment's home directory into other directories unless approved by the DII COE Chief Engineer. (This requirement is stipulated to avoid circumventing the intent of the *I&RTS* by loading the segment as directed by the installer, and then moving the segment to some other location during `PostInstall`.) This requirement does *not* apply to COTS segments, to patch segments - as approved by the Chief Engineer - that must move files into the segment being patched, nor does it apply to data that is being moved to the proper `$DATA_DIR/global` or `$DATA_DIR/local` directory.
T	**F**	**N/A**	**5-14**	(NT) The segment creates all its subkeys underneath *`SegType`*`\`*`SegDirName`* where *`SegType`* is `Account Groups`, `COE`, `COTS`, `Patches`, `Data`, or `Software`, and *`SegDirName`* is the segment's directory name.
T	**F**	**N/A**	**5-15**	(NT) Unless a COTS segment, the segment does not create any root keys.
T	**F**	**N/A**	**5-16**	(NT) All segment subkeys are named with the segment prefix.
T	**F**	**N/A**	**5-17**	(NT) The segment supports UNC filenames.
Operating System				
T	**F**	**N/A**	**5-18**	The segment does not rename well defined ports (e.g., `ftp`, `listen`), or declare new port names which have the same port number as well-defined ports in the UNIX `/etc/services` file, or the NT equivalent of this file.
T	**F**	**N/A**	**5-19**	If ports are required, they have been identified and documented in the `COEServices` segment descriptor.
GUI Environment				
T	**F**	**N/A**	**5-20**	The segment is fully compliant with the style of the native GUI (see compliance requirements in the *DII User Interface Specification*).
T	**F**	**N/A**	**5-21**	The segment uses the window manager provided by the COE (`dtwm`[4] for UNIX, Windows NT for NT platforms).
T	**F**	**N/A**	**5-22**	(UNIX) The segment is compatible with the `XFONTSDIR`, `XAPPLRESDIR`, and `XENVIRONMENT` settings established by the COE. A segment may change the settings of these environment variable s as long as they are in effect *only* for the segment's local environment and do not affect the global execution environment.

[4] With the present *I&RTS* release, a commercial CDE product provides the desktop. Thus, `dtwm` replaces `mwm` from the previous *I&RTS*. There should not be any impact to any segment that presently works under `mwm`.

Database Services				
T	**F**	**N/A**	**5-23**	Application segments are separate from their corresponding database segment.
T	**F**	**N/A**	**5-24**	Application segments that access databases operate correctly from any COE-compliant platform and are not required to be installed on a database server.
T	**F**	**N/A**	**5-25**	Segments are not tied to a particular server name (i.e., The segment does not hardcode a server name.)
T	**F**	**N/A**	**5-26**	The segment installation revokes the owner account's DBMS login privilege upon successful completion of database installation so that no owner accounts can be used to connect to the database.
T	**F**	**N/A**	**5-27**	Owner accounts are not used to connect to databases except during segment installation.
T	**F**	**N/A**	**5-28**	Database owner accounts do not have database administrator privileges.
T	**F**	**N/A**	**5-29**	Separate segments are provided to create required database dependencies. These segments are executed by the owning database(s).
T	**F**	**N/A**	**5-30**	The segment installation requires the owner account password to be changed upon completion.
T	**F**	**N/A**	**5-31**	Segments do not modify the core DBMS instance's configuration.
T	**F**	**N/A**	**5-32**	The segment does not assume any particular disk configuration when creating data files.
T	**F**	**N/A**	**5-33**	Any modified versions of DBMS COE tools reside with the application's client segment.
T	**F**	**N/A**	**5-34**	Scripts are provided for the DBA's use to add, modify, and remove user privileges. These scripts are documented and the documentation is submitted to the SSA with the segment.
T	**F**	**N/A**	**5-35**	The segment does not modify another segment's database schema.
T	**F**	**N/A**	**5-36**	Grants are not made to public or general-purpose users (e.g. Oracle's `PUBLIC` user).
T	**F**	**N/A**	**5-37**	Only the owner and the DBA are able to administer grants.
T	**F**	**N/A**	**5-38**	Operations that set or redirect the user's DBMS environment variable s take place only within the application's execution space.
T	**F**	**N/A**	**5-39**	No indices are created on another segment's database tables.
T	**F**	**N/A**	**5-40**	Application-level permissions are not granted to DBA accounts or to database roles used for DBMS administration.

T	**F**	**N/A**	**5-41**	Database segments are identified as universal, unique, or sharable according to their potential for sharing.
Web Services				
T	**F**	**N/A**	**5-42**	The segment's HTML files are in the segment's `$DATA_DIR/local/`*`SegDir`*`/pub` directory.
T	**F**	**N/A**	**5-43**	The segment supports HTML 3.2 and complies wit h style specifications (see the *DII User Interface Specification*) for Web applications.
T	**F**	**N/A**	**5-44**	The segment provides a notification to "disadvantaged" users if they are using a browser that does not support the features provided by the segment.
Runtime Environment				
T	**F**	**N/A**	**5-45**	The segment is launched from the same desktop provided with the COE.
T	**F**	**N/A**	**5-46**	The desktop is configured in accordance with the *DII User Interface Specification*.
T	**F**	**N/A**	**5-47**	The segment uses relative pathname s for files within the segment.
T	**F**	**N/A**	**5-48**	The segment does not use the "~" character, for UNIX or its NT equivalent, for referencing pathnames in environment extension files which become a part of the global runtime environment.
T	**F**	**N/A**	**5-49**	The segment does not alter any reserved symbols from the *I&RTS* Chapter 5 unless authorized to do so by the DII COE Chief Engineer.
T	**F**	**N/A**	**5-50**	The segment does not override or alter any environment variable that it doesn't create. The segment may extend environment variables set by the affected account group through environment extension files .
T	**F**	**N/A**	**5-51**	The segment completely separates the development environment from the runtime environment, and no development environment tools, scripts, or other executables are required at runtime.
T	**F**	**N/A**	**5-52**	The segment uses the same global runtime environment configuration as provided by the COE; extensions, if any, are made through the appropriate environment extension files and segment descriptors.
T	**F**	**N/A**	**5-53**	The segment only listens on assigned ports, only registers assigned RPC addresses, and only adds assigned system UIDs (UNIX).
T	**F**	**N/A**	**5-54**	The segment home environment variable points to the segment's home directory. The name of the environment variable is *`segprefix`*`_HOME` where *`segprefix`* is the segment's assigned prefix.

T F N/A	**5-55**	(UNIX) The segment uses environment extension files to add no more than one “home” environment variable to the affected account group. (This then becomes part of the “global” execution environment. It is preferable that the segment use only “local” home environment variables, if any.)
COE-Component Segments		
T F N/A	**5-56**	The segment has been authorized as a COE-component segment by the DII COE Chief Engineer.
T F N/A	**5-57**	Segments in the COE kernel fully specify dependencies upon supporting components within the kernel. This is done through the `Requires` segment descriptor.

T F N/A	**5-58**	If a parent COE-component segment, the following environment variable s (as appropriate for a UNIX versus NT environment) are automatically defined as specified by this document:
		`DATA_DIR`
		`LD_LIBRARY_PATH`
		`LOGNAME`
		`LOG_NAME`
		`LOGIN_NAME`
		`MACHINE_CPU`
		`MACHINE_OS`
		`path`
		`TMPDIR`
		`TZ`
		`XAPPLRESDIR`
		`XFONTSDIR`
		`XENVIRONMENT`

T F N/A	**5-59**	If a COE child segment, the segment does not alter the UNIX `path` environment variable . (This rule does not apply to the parent COE-component segment.)
T F N/A	**5-60**	All executables and other public symbols use the segment prefix unless otherwise approved by the DII COE Chief Engineer . (Certain legacy segments may be “grandfathered” by the DII COE Chief Engineer.)
Account Groups		
T F N/A	**5-61**	If the account group is part of the COE, prior approval has been received from the DII COE Chief Engineer to create an account group segment.
T F N/A	**5-62**	If the account group is a mission-application segment, p rior approval has been received from the Chief Engineer to create an account group segment.

T	**F**	**N/A**	**5-63**	The environment settings from `/h/COE/Scripts` are automatically included in the runtime environment of the account group being created. (In UNIX this may be accomplished by "sourcing" `/h/COE/Scripts/.cshrc.COE`.)
T	**F**	**N/A**	**5-64**	The segment provides an executable in the `Scripts` subdirectory, named `Run`*`segprefix`* where *`segprefix`* is the segment prefix, to initiate execution of the account group's applications.

T	**F**	**N/A**	**5-65**	The following environment variables, as appropriate for NT versus UNIX, are defined:
				`COE_SYS_NAME`
				`DISPLAY`
				`HOME`
				`path`
				`SHELL`
				`TERM`
				`USER`
				`USER_HOME`
				`USER_DATA`
				`USER_PROFILE`

T	**F**	**N/A**	**5-66**	(UNIX) The segment provides files of the form `filename.`*`segprefix`* for all environment files that segments may reference or extend through the `ReqrdScripts` descriptor.
T	**F**	**N/A**	**5-67**	(NT) The segment establishes any required global environment settings in the registry.
Aggregate Segments				
T	**F**	**N/A**	**5-68**	If a parent segment, the segment does not specify a dependency on any of its child segments.
T	**F**	**N/A**	**5-69**	If a child segment, the segment does not specify a dependency on its parent segment nor any other children in the aggregate.
T	**F**	**N/A**	**5-70**	Only one segment in the aggregate is designated as the parent.
Segment Descriptors				
T	**F**	**N/A**	**5-71**	The segment uses `SegInfo` or individual segment descriptor files, but not both.
T	**F**	**N/A**	**5-72**	The segment describes all background processes, if any, through the `Processes` descriptor.
T	**F**	**N/A**	**5-73**	All segment dependencies and conflicts are fully declared through the appropriate descriptor. (Mission-application segments need not specify dependencies on segments contained in the COE kernel unless they are version sensitive. COE-component segments need not specify dependencies on the COE kernel unless they are sensitive to version changes in the COE kernel.)

T	**F**	**N/A**	**5-74**	Memory and disk space requirements are fully and accurately specified in the `Hardware` descriptor.
T	**F**	**N/A**	**5-75**	If not a permanent segment, the `DEINSTALL` script and `Comm.deinstall` descriptor have been fully tested to ensure they correctly make the changes indicated and completely restore the system to the state it was in prior to loading the segment.
T	**F**	**N/A**	**5-76**	The segment `Community` and `Comm.deinstall` (if applicable) descriptors have been fully tested to ensure that they correctly makes the changes indicated, and that they do not inadvertently destroy settings that may have been made by another segment. [5]
T	**F**	**N/A**	**5-77**	The `ReqrdScripts` descriptor contains no more than 32 script names and no script name is longer than 32 characters.
T	**F**	**N/A**	**5-78**	(UNIX) The `PostInstall`, `PreInstall`, and `DEINSTALL` scripts have been checked and verified to *not* do a UNIX `mv` across file partitions.
T	**F**	**N/A**	**5-79**	(NT) Unless a COTS segment, the segment uses the `Processes` descriptor to create boot time processes. It does not set the`Run` or `RunOnce` keys underneath `CurrentVersion`.
T	**F**	**N/A**	**5-80**	(NT) The segment's executable descriptors use the `.EXE` extension for compiled executables and `.BAT` for batch files.
T	**F**	**N/A**	**5-81**	(NT) The segment uses `SegInfo` and *not* individual segment descriptor files.
Process Compliance				
T	**F**	**N/A**	**5-82**	The segment has been registered with the SSA.
T	**F**	**N/A**	**5-83**	The cognizant DOD Chief Engineer has granted prior approval for background, boot, RunOnce, and periodic processes.
T	**F**	**N/A**	**5-84**	System resources required by the segment have been registered with the SSA.
T	**F**	**N/A**	**5-85**	The segment prefix being used is the prefix assigned at segment registration time.
T	**F**	**N/A**	**5-86**	The ports, UIDs (UNIX), and RPC addresses being used are those assigned at segment registration time.
T	**F**	**N/A**	**5-87**	The platforms and operating systems on which the segment can run have been identified and documented in a *Version Description Document*, or its equivalent.
T	**F**	**N/A**	**5-88**	All COTS products required, including the required version, are documented in the *Version Description Document* or its equivalent.

[5] Developers should generally use `$APPEND` to add to community files, rather than `$DELETE` or `$REPLACE`. Developers should ensure that they delete or replace only those entries to a community file that their segment would have added.

T	**F**	**N/A**	**5-89**	All required licenses are provided to the SSA with the segment, or negotiations have been made with the SSA to use licenses procured by the SSA.
T	**F**	**N/A**	**5-90**	Segment dependencies are noted in the *Version Description Document* or its equivalent.
T	**F**	**N/A**	**5-91**	The *Version Description Document*, or its equivalent, has been submitted with the segment to the SSA.
T	**F**	**N/A**	**5-92**	The segment has been submitted to and a ccepted for inclusion in the SSA's on line library.
T	**F**	**N/A**	**5-93**	The `VERSION` descriptor has been updated from the previous release in accordance with the requirements specified in Chapter 5. (This does not apply to the initial release of the segment.)
T	**F**	**N/A**	**5-94**	The segment is submitted with an annotated output from `VerifySeg`. All warnings are explained in full in `VSOutput`.[6]
T	**F**	**N/A**	**5-95**	The segment is submitted with a set of integration notes (`IntgNotes`) as described in Chapter 5.
T	**F**	**N/A**	**5-96**	The segment has been loaded and tested in the COE environment prior to submission to the SSA.
T	**F**	**N/A**	**5-97**	Segment installation has been tested through the same installation tools used by site operators. (`TestInstall` alone does *not* satisfy this requirement. The `COEInstaller` tool must be used to load and remove the segment.)
T	**F**	**N/A**	**5-98**	If removable, the segment has been tested and confirmed that it can be successfully removed from the system.
T	**F**	**N/A**	**5-99**	If special installation/integration procedures/problems exist, then they are incorporated into the `PostInstall` (or other) descriptors as appropriate, and documented in the `IntgNotes` descriptor file.
T	**F**	**N/A**	**5-100**	(NT) Unless a COTS segment or authorized by the DII COE Chief Engineer , the segment does not register "uninstall" information in the registry (e.g., subkey `CurrentVersion\Uninstall`). ("uninstall" information is handled automatically for the segment by the COE installation tools.)
T	**F**	**N/A**	**5-101**	(NT) If an approved segment registers "uninstall" information in the registry , the `$USES_UNINSTALL` keyword is declared in the segment's `Direct` descriptor.
Miscellaneous				
T	**F**	**N/A**	**5-102**	The segment creates and initializes dynamic data files that are updated as the system executes (e.g., message logs, operator preferences). If an expected file is missing, the segment generates a runtime error message and gracefully terminates with an appropriate message to the operator.

T	**F**	**N/A**	**5-103**	If a patch segment, it follows the patch segment naming convention.
T	**F**	**N/A**	**5-104**	The segment does not alter *any* files outside its own directory with the following exceptions: (a) the segment is a patch segment and must modify files in another segment; (b) the segment is creating temporary files or directories in directories established for temporary storage; (c) the segment is modifying files created for it by the operating system; (d) the files are created or modified through approved APIs or segment descriptors during segment installation; or (e) the files are owned by the segment but reside in approved COE locations outside the segment's assigned directory (e.g., /h/data/local, /h/data/global, user directories).
T	**F**	**N/A**	**5-105**	The segment does not create copies of executables from other segments. (There are rare instances where this may be required to create a patch segment. Such exceptions require the prior approval of the Chief Engineer.)
T	**F**	**N/A**	**5-106**	The segment does not contain any circular dependencies (e.g., Seg A depends on Seg B, Seg B depends on Seg C, Seg C depends on Seg A is not allowed).
T	**F**	**N/A**	**5-107**	The segment does not delete itself via the `DEINSTALL` descriptor, nor perform any other operations that are handle d by the COE installation tools (e.g., undo changes made to community files).

[6] This can be done by redirecting the output of `VerifySeg` to the file `VSOutput`. Then, use any convenient ASCII editor to edit `VSOutput` to insert comments to explain all warning messages.

Intermediate DII Compliance (Level 6)

Security				
T	**F**	**N/A**	**6-1**	The segment satisfies at least one of the following two requirements: (1) The segment contains only subdirectories directly underneath the segment 's home directory. All files are at least one level down from the segment's home directory. (2) The segment has no directories or files that have the equivalent of the UNIX 777 file permissions.
T	**F**	**N/A**	**6-2**	If the data for a particular segment contains any classified entries, then all of its data is packaged in a separate data segment and classified accordingly.
T	**F**	**N/A**	**6-3**	Classified applications segments are packaged separately from unclassified segments, or from segments which are classified at a lower level. (It is permissible to create aggregate segments that contain segments at different classification levels, but the aggregate must be labeled with the highest classification level of any segment within the aggregate.)
T	**F**	**N/A**	**6-4**	Termination of segment execution, whether premature, inadvertent, or intentional does not place the operator at a command-line prompt.
T	**F**	**N/A**	**6-5**	The Chief Engineer has authorized privileged processes in the segment. The `$PRIVPROC` keyword is stated in the `Direct` segment descriptor, and the privileged processes are listed in the `Processes` segment descriptor.
T	**F**	**N/A**	**6-6**	(UNIX) The segment does not contain any shell scripts that `SUID` or `SGID` to root.
Standards Compliance				
T	**F**	**N/A**	**6-7**	The segment is either completely comp liant with the *DII User Interface Specification* or has minimal deviations that have been approved by the Chief Engineer .
T	**F**	**N/A**	**6-8**	The segment is available on all COE-supported platforms unless otherwise approved by the Chief Engineer .
T	**F**	**N/A**	**6-9**	The segment does not alter any community files except through COE segment descriptors or published APIs.
T	**F**	**N/A**	**6-10**	The segment does not use directories with different names than specified in Chapter 5 to fulfill the purpose of `Scripts`, `bin`, `data`, etc. (`progs` and `libs` are acceptable for this level for as long as the COE tools support them.)
T	**F**	**N/A**	**6-11**	If the segment contains APIs written in C, the header files for the public APIs are ANSI-C-compliant and use function prototypes, and the header files are constructed to support C++ calling routines as described in Chapter 9.
T	**F**	**N/A**	**6-12**	Specification files for Ada are included for all APIs, unless the Chief Engineer has granted a waiver.

T	**F**	**N/A**	**6-13**	(NT) All `INI` files used that are local to the segment are stored in the segment's `data/INI` subdirectory.
T	**F**	**N/A**	**6-14**	(NT) The segment supports long filenames.
T	**F**	**N/A**	**6-15**	(NT) The segment uses filename extensions in accordance with standard Windows usage (`TXT` for ASCII files, `DLL` for dynamic link libraries, etc.).
GUI Environment				
T	**F**	**N/A**	**6-16**	(UNIX) The segment does not alter any X or Motif supplied files (e.g., `Xdefaults`, `rgb.txt`).
Database Services				
T	**F**	**N/A**	**6-17**	Neither Informix nor Oracle Public Synonyms are used.
T	**F**	**N/A**	**6-18**	Database segments do not create user accounts, except for a database services account.
T	**F**	**N/A**	**6-19**	Grants are made to database roles/groups, not user accounts or general-purpose users (e.g., Oracle's `PUBLIC` user).
T	**F**	**N/A**	**6-20**	The application does not assume the existence of any particular user.
T	**F**	**N/A**	**6-21**	Data elements do not use machine-dependent data types.
T	**F**	**N/A**	**6-22**	The segment does not create data objects in other segments except through documented inter-database dependencies (e.g., triggers) and published APIs.
T	**F**	**N/A**	**6-23**	External object dependencies are listed under the `Database` descriptor.
T	**F**	**N/A**	**6-24**	The segment uses only the DBMS provided by the COE, or has an approved migration plan.
T	**F**	**N/A**	**6-25**	The segment either implements DOD 8320 data standards, or has an approved plan for doing so. (The migration plan must be coordinated with the DII COE Chief Engineer for any data fields that are part of Universal or Shared data segments. Data fields that are part of a Unique data segment do not require DII COE Chief Engineer approval.)
T	**F**	**N/A**	**6-26**	Database roles that span multiple database segments are defined in their own segments.
T	**F**	**N/A**	**6-27**	Data objects and elements follow naming conventions specified in Chapter 4.
T	**F**	**N/A**	**6-28**	Definitions for schema components are provided in the DBMS data dictionary.

Web Services				
T	**F**	**N/A**	**6-29**	The segment uses the Web server provided by the COE rather than bringing along its own Web server.
COTS Products				
T	**F**	**N/A**	**6-30**	All COTS products are packaged as separate, individual COTS segments.
T	**F**	**N/A**	**6-31**	The `PostInstall` or `PreInstall` script ensures that there is enough space in the directories where the COTS product will be installed and uses `COEInstError` to report an error message if not.
T	**F**	**N/A**	**6-32**	The `FilesList` descriptor has been validated as correctly documenting what files and directories constitute the COTS product. This does not apply to COTS products in the COE kernel.
Runtime Environment				
T	**F**	**N/A**	**6-33**	If the segment creates temporary files, they are deleted when no longer needed.
T	**F**	**N/A**	**6-34**	If the segment uses absolute pathnames to reference files outside the segment, it is able to determine the absolute path at runtime, and is able to handle symbolic links that are themselves symbolic links.
T	**F**	**N/A**	**6-35**	The segment reuses environment variable s already defined by the COE or by the affected account group. It does not create any environment variables that are identical in value to those defined by the COE or the affected account group, or that can be derived from them.
T	**F**	**N/A**	**6-36**	The segment does not create any environment variable s or other public symbols with the same name as any environment variables listed as reserved in the *I&RTS*.
T	**F**	**N/A**	**6-37**	Shared libraries (UNIX) and DLLs (NT) provided by the segment are in the segment's `bin` subdirectory. The `SharedFile` descriptor is used to define them, and they are named using the segment prefix convention.
T	**F**	**N/A**	**6-38**	The segment does not insert the current working directory (e.g., ".") into the search path for executables.
T	**F**	**N/A**	**6-39**	If the segment provides public APIs, all uses of signals and process or thread creation within the segment's public libraries are documented in the appropriate programmer's guides. Moreover, all such API functions shall be reentrant to allow them to be called from a multithreaded application.
T	**F**	**N/A**	**6-40**	(UNIX) The global environment is extended through runtime extension files in the segment 's `Scripts` subdirectory.
T	**F**	**N/A**	**6-41**	(UNIX) Fonts and app-defaults located underneath the segment 's `data` subdirectory follow the segment prefix naming convention specified in the *I&RTS*.

T	**F**	**N/A**	**6-42**	(UNIX) The segment appends, not prepends, its `bin` subdirectory to the environment variable used for the search path for finding executables. (This does not apply to COE child segments.)
T	**F**	**N/A**	**6-43**	(UNIX) The segment uses relative pathnames or symbolic links to reference files within the segment.
T	**F**	**N/A**	**6-44**	(NT) The segment uses relative pathnames or shortcuts to reference files within the segment.
T	**F**	**N/A**	**6-45**	(NT) The segment stores its DLL files in the segment's `bin` subdirectory.
T	**F**	**N/A**	**6-46**	(NT) Unless a COTS segment, the segment does not alter the Windows `path` environment variable .
T	**F**	**N/A**	**6-47**	(NT) The segment does not use polling as a synchronization technique.
T	**F**	**N/A**	**6-48**	(NT) The segment does not use MS-DOS functions.
Segment Descriptors				
T	**F**	**N/A**	**6-49**	The `ReleaseNotes` descriptor conforms to the requirements stipulated in Chapter 5.
T	**F**	**N/A**	**6-50**	If any files need special permission/ownership settings, they are established through the `FileAttribs` descriptor if the descriptor supports the required setting. Exceptions to this are documented and approved by the Chief Engineer.
Process Compliance				
T	**F**	**N/A**	**6-51**	The segment includes an API test suite that exhaustively exercises all APIs provided by the segment.
T	**F**	**N/A**	**6-52**	The segment includes man pages , or HTML-format pages, for all APIs that are to be distributed with the Developer's Toolkit.
T	**F**	**N/A**	**6-53**	The segment has been compiled without the debug option enabled .
T	**F**	**N/A**	**6-54**	If the segment has published APIs implemented as shared libraries, static libraries are provided as well.
T	**F**	**N/A**	**6-55**	If the segment uses another segment's public APIs and they are implemented as shared libraries, the segment is submitted linked with the shared libraries and not the static libraries.
T	**F**	**N/A**	**6-56**	If the segment has a `DEINSTALL` and `Community` descriptor, it also includes a `Comm.deinstall` descriptor which reverses the actions of the `Community` descriptor during segment removal.
T	**F**	**N/A**	**6-57**	The segment has been tested to ensure that it successfully installs over and replaces any previous version of the segment.
T	**F**	**N/A**	**6-58**	If the segment contains a large static database, it is provided as a separate data segment.

T	**F**	**N/A**	**6-59**	(UNIX) The segment executables have been run through the UNIX `strip` program.
Miscellaneous				
T	**F**	**N/A**	**6-60**	Unless an account group segment, the segment is integrated within one or more of the predefined account groups.
T	**F**	**N/A**	**6-61**	If the COE provides functions required by the segment, at least 50% of the functions required are provided by the COE and not by duplicative code in the segment.
T	**F**	**N/A**	**6-62**	API backwards compatibility conforms to the version numbering scheme described in Chapter 3.
T	**F**	**N/A**	**6-63**	The segment does not provide access to a command-line prompt, except with prior Chief Engineer approval.

Interoperable Compliance (Level 7)

Security		
T F N/A	**7-1**	The segment does not place any temporary files in the system maintained temporary directory that are sensitive to alteration, deletion, or disclosure to unauthorized users.
T F N/A	**7-2**	If the segment creates files that are sensitive to alteration or deletion by unauthorized users, they are not placed in any directory where such users have write access, and those files do not have write permissions set for such users.
T F N/A	**7-3**	If the segment creates files which are sensitive to disclosure to unauthorized users, they are not placed in any directory where users have such access.
T F N/A	**7-4**	Entering a command-line mode requires the operator to enter a password and forces execution of the system login process.
T F N/A	**7-5**	The segment does not contain features with multiple security levels, unless an aggregate segment.
T F N/A	**7-6**	Unclassified sample data is provided with the segment to allow for unclassified testing and training.
T F N/A	**7-7**	The segment does not create files or directories with write permissions for “world” users, except as authorized by the Chief Engineer.
T F N/A	**7-8**	Data files with different file permissions are split into separate directories underneath the segment’s `data` subdirectory.
Standards Compliance		
T F N/A	**7-9**	If written in C, the segment is ANSI-C-compliant.
T F N/A	**7-10**	If written in Ada, the segment is Ada-95-compliant unless otherwise authorized by the Chief Engineer.
T F N/A	**7-11**	If the segment contains public APIs, Ada and C interfaces are both provided unless the Chief Engineer grants a waiver.
T F N/A	**7-12**	Global and local data owned by the segment are located underneath `$DATA_DIR` as described in Chapter 5.
T F N/A	**7-13**	Operator-specific data is located underneath `/h/USERS` as described in Chapter 5.
T F N/A	**7-14**	Excepting COTS segments, all environment variables are named with the segment prefix unless approved by the Chief Engineer. (The Chief Engineer may authorize “grandfathering” of certain environment variables.)
T F N/A	**7-15**	The segment uses only POSIX.1-defined interfaces to access the operating system, unless authorized by the Chief Engineer.

				Network Services
T	**F**	**N/A**	**7-16**	(NT) The segment determines the location for shared data through the registry .
T	**F**	**N/A**	**7-17**	(NT) The segment stores information about shared resources in the location specified in Chapter 6.
				GUI Environment
T	**F**	**N/A**	**7-18**	The segment uses resource files to control window behavior rather than hardcoded window behavior attributes.
T	**F**	**N/A**	**7-19**	The segment supports cut and paste between GUI-based segments through the use of a shared clipboard.
T	**F**	**N/A**	**7-20**	(NT) The segment uses TrueType fonts.
				Database Services
T	**F**	**N/A**	**7-21**	Data objects within the segment do not duplicate those already contained in available Universal database segments.
T	**F**	**N/A**	**7-22**	Database fragmentation schemas are contained in separate segments.
T	**F**	**N/A**	**7-23**	Database roles/groups are specific to application privileges, not general purpose.
T	**F**	**N/A**	**7-24**	The segment does not duplicate any data available from the SHADE repository, except for performance reasons, unless approved by the DII COE Chief Engineer.
T	**F**	**N/A**	**7-25**	The segment uses only FIPS-127-2 SQL-defined interfaces to access the RDBMS query services.
T	**F**	**N/A**	**7-26**	Data object creation script files follow the specified structure and naming convention.
T	**F**	**N/A**	**7-27**	The data objects contained within a database segment are standardized according to DOD 8320 guidance.
T	**F**	**N/A**	**7-28**	All constraints and business rules are in the database, not the applications.
T	**F**	**N/A**	**7-29**	The database server segment provides a reload capability and a non-destructive update capability.

				DCE Services
T	**F**	**N/A**	**7-30**	If the application uses DCE services, only the DCE interfaces defined by the DCE version supported by the COE (see Appendix A) are used to access those services.
				Runtime Environment
T	**F**	**N/A**	**7-31**	The segment does not include any environment variable s that could be derived from an already def ined environment variable.
T	**F**	**N/A**	**7-32**	Segment references to global and local data are done through the `$DATA_DIR` environment variable .
T	**F**	**N/A**	**7-33**	(NT) The segment stores private INI files, if any, in the segment's `data\INI` subdirectory.
				Miscellaneous
T	**F**	**N/A**	**7-34**	The segment does not duplicate any functions provided by COE-component segments unless appro ved by the DII COE Chief Engineer.
T	**F**	**N/A**	**7-35**	No more than 25% of the segment's accesses to COE-component segments is through private APIs.
T	**F**	**N/A**	**7-36**	(NT) The segment does not duplicate any Windows functions.

Full DII Compliance (Level 8)

Security				
T	**F**	**N/A**	**8-1**	Entry to and exit from the command-line mode causes an entry into the system audit logs that specifies the date, time, and user involved.
T	**F**	**N/A**	**8-2**	Information written to the audit log includes the segment prefix.
T	**F**	**N/A**	**8-3**	The segment does not mix restricted and unrestricted data files in the same directory.
Standards Compliance				
T	**F**	**N/A**	**8-4**	The segment does not use any conventions obsoleted by this document (use of `progs` vs. `bin`, use of `COMPONENT` vs. `CHILD`, use of `ModName` and `SegType` vs. `SegName` etc.).
T	**F**	**N/A**	**8-5**	All public symbols are named with the segment prefix naming convention.
T	**F**	**N/A**	**8-6**	All directory and file names begin with an alphanumeric character.
T	**F**	**N/A**	**8-7**	The segment follows the convention that data owned by the segment under `$DATA_DIR` is in the form `$DATA_DIR/local/`*`segdir`*`/data` and `$DATA_DIR/global/`*`segdir`*`/data` where *`segdir`* is the segment's home directory name.
GUI Environment				
T	**F**	**N/A**	**8-8**	The segment is fully compliant with the *DII User Interface Specification*.
T	**F**	**N/A**	**8-9**	(NT) The segment uses common control and dialog functions from `COMCTL32.DLL` and `COMDLG32.DLL`.
T	**F**	**N/A**	**8-10**	(NT) The segment properly handles the window close message.
T	**F**	**N/A**	**8-11**	(NT) The segment uses the Windows print dialog box for selecting printer configuration parameters.

Database Services				
T	**F**	**N/A**	**8-12**	Data elements are chosen from Joint standards and use the data type, field width, and units of measure prescribed in the standard.
T	**F**	**N/A**	**8-13**	A test database is provided together with test procedures to verify correct installation of the database and associated roles, and to verify correct operation of constraints defined in the database.
T	**F**	**N/A**	**8-14**	The segment does not duplicate any data already maintained in the SHADE repository *or* the COE-based target system, unless for performance reasons and only as approved by the DII COE Chief Engineer.
T	**F**	**N/A**	**8-15**	The segment uses only the DBMS provided by the COE.
Runtime Environment				
T	**F**	**N/A**	**8-16**	The segment adds no more than one "home" environment variable to the global environment.
T	**F**	**N/A**	**8-17**	All executables and public symbols are named `segprefix_name`, where `segprefix` is the assigned segment prefix.
T	**F**	**N/A**	**8-18**	(NT) Local environmental settings are established through an `LOCALENV.BAT` file in the segment's `Scripts` subdirectory.
Segment Descriptors				
T	**F**	**N/A**	**8-19**	The segment uses `SegInfo` rather than individual segment descriptor files.
Process Compliance				
T	**F**	**N/A**	**8-20**	The segment includes a set of test data for verifying correct segment operation.
Miscellaneous				
T	**F**	**N/A**	**8-21**	The segment does not use any private APIs to access external segments. All a ccesses are through public APIs or approved protocol standards.
T	**F**	**N/A**	**8-22**	Operator data is located through the *Preferences* APIs.
T	**F**	**N/A**	**8-23**	The current operator profile is obtained through the *Preferences* APIs.
T	**F**	**N/A**	**8-24**	The segment does not duplicate functionality provided by any other segment unless approved by the DII COE Chief Engineer.

Recommended Guidelines

The items contained in the following checklist are not mandatory, but are general guidelines for most segments. They are not considered in establishing the DII compliance level.

T	**F**	**N/A**	**M-1**	The segment does not use symbolic links.
T	**F**	**N/A**	**M-2**	The segment does not use boot or background processes. It uses session-or-transient-level processes instead.
T	**F**	**N/A**	**M-3**	The segment allows comments in ASCII data files. The # character is the standard for single line comments while C style comments[7] (delimited by the /* */ pair) are the standard for all other comments.
T	**F**	**N/A**	**M-4**	If written in C, the segment has been run through `lint` to detect potential coding errors.
T	**F**	**N/A**	**M-5**	If written in C, the segment has been compiled with the `STRICT` constant.
T	**F**	**N/A**	**M-6**	The segment uses `putenv` or an equivalent technique to create segment-specific environment variables that are inherited locally, rather than adding environment variables to the global environment.
T	**F**	**N/A**	**M-7**	(UNIX) The segment links with X and Motif shared libraries.
T	**F**	**N/A**	**M-8**	(NT) The segment uses Microsoft DLLs for required functions that are provided by NT.
T	**F**	**N/A**	**M-9**	(NT) The segment links to DLL functions by using symbolic names, not ordinal numbers.
T	**F**	**N/A**	**M-10**	(NT) The segment exports DLL functions by symbolic name, not ordinal numbers.

[7] C style comments for ASCII data files are recommended over Ada-style comments, not because of a preference for C over Ada, but because it allows an entire block of lines to be easily commented out. Ada-style comments would require editing each individual line to insert the "--" comment delimiter.

T	**F**	**N/A**	**M-11**	(NT) The segment does not include an `app-defaults` or `fonts` subdirectory.
T	**F**	**N/A**	**M-12**	(NT) The segment uses the Win32 API GDI for creating 2D graphics.
T	**F**	**N/A**	**M-13**	(NT) The segment uses OpenGL APIs for 3D graphics.
T	**F**	**N/A**	**M-14**	(NT) `PORTTOOL.EXE` has been used to identify potential problems with how the segment uses Windows APIs.
T	**F**	**N/A**	**M-15**	(NT) The segment operates correctly under both Windows NT and Windows 95.
T	**F**	**N/A**	**M-16**	(NT) The segment uses message crackers from `WINDOWSX.H`.

This page is intentionally blank.

Appendix C: COE Tools

This appendix lists the COE tools available to assist developers in creating, installing, and testing segments. This appendix is not intended to provide an exhaustive discussion of all tool features. Additional features and improvements in the tools may be made before updates to this document are released.

The previous *I&RTS* provided more detailed information on parameters the tools accept, and their syntax. This level of detail has been removed from this version of the *I&RTS* and moved to Developer’s Toolkit documentation. Backwards compatibility has been preserved, but there are new tools with this release, and some of the tools accept additional parameters to provide new functionality. Refer to the Developer's Toolkit release notes, the online man pages for the tools, and the help page provided by the tools for the latest information.

General Tool Features

Most of the tools are designed to run interactively, and accept one or more command-line parameters. Many of the tools are designed to be used in the installation process during `PostInstall` or `PreInstall`, while others are intended to be used during the development process. The development tools may be invoked individually from a command line, or from an Integrated Development Environment tool, `COEIDE`.

Tools that begin with the prefix `COE` are runtime tools[8] and are thus distributed to operational sites. Those which do not begin with the prefix `COE` are developer tools only and are distributed as part of the Developer's Toolkit, but are not normally sent to operational sites. This section describes other conventions that are common to the tools.

Communicating with the Tools

The COE tools can be accessed in two different ways. First, a user program can be written and linked with the tools library.[9] Such a program goes through the defined APIs to request services from the tools. A second way that the tools can be used is to execute the tools as a separate program and use services provided by the operating system to communicate between the tool and the calling program. This technique is most frequently used in as a way to invoke the tools from within a `PostInstall` script. Whichever approach is most convenient may be used. All of the tools are available as individual executables, but not all are available through a library interface.

When accessed through a library interface, the tools return data through the interface. They may also write to `stdout`. When used as separate executable programs, the tools communicate with the outside world in two ways.

1. The tools use the `exit` function to set the UNIX `status` environment variable. (`status` is used for the C shell `csh`. The `$?` environment variable is used for POSIX shells.) `status` is set to 0 for normal tool completion. A `status` return value other than 0 indicates a completion code that is tool-specific. Refer to the appropriate Developer's Toolkit documentation on man pages for a complete description of the tools and their return codes, including error return codes.[10]

2. The tools use `stdin`, `stdout`, and `stderr` and thus support I/O redirection. Redirecting `stdin` allows the tools to receive input from a file or another program, while redirecting `stdout` allows the tools to provide input to other programs. (Redirecting `stdin` is not always convenient. The `-R` command-line parameter, see

[8] The one exception to the convention of using the prefix `COE` for runtime tools is `COEIDE`. `COEIDE` is intended only for the development process and would not normally be distributed to operational sites.
[9] Libraries for NT platforms are implemented as DLLs (Dynamic Link Libraries).
[10] The previous *I&RTS* indicated that the tools return a value of `-1` to indicate an error. This convention has been preserved to the extent possible, but note that certain operating systems return a single byte as the `status` environment variable and hence return a value of 255 instead of `-1`. Refer to the appropriate Developer's Toolkit documentation or online man pages for complete documentation of tool return codes.

below, allows a tool to read input from a response file instead of `stdin`.) For example, the tool `COEPrompt` displays a message to the user and allows the user to type in a response, and the user's response is written to `stdout`. The tools write normal output to `stdout`, while errors and warnings are sent to `stderr`.

For UNIX, the following statement illustrates how the second technique can be used to ask the user to enter the name of a file:

```
COEPrompt "Enter Filename" | MyProg
```

Or, the following can be used to write the results to a file:

```
COEPrompt "Enter Filename" >& /tmp/tempfile
```

Note that the above example redirects both `stdout` and `stderr` to `/tmp/tempfile`.

In the NT environment,

```
COEPrompt “Enter Filename” 1> C:\temp\tempfile 2>1&
```

will redirect output to the file `C:\temp\tempfile`.

Standard Parameters

The command-line parameters listed below are common to all tools for which they are meaningful:

`-C file`	Read command-line parameters from *file*.
`-h, H`	Display online help describing how to use the tool.
`-p path`	Use *path* to establish the path for subsequent file names.
`-R file`	Use *file* to respond to questions from the tool.
`-v`	Toggle the "verbose" flag. When enabled, print out diagnostic information as the tool executes. The initial state of the flag is disabled.
`-V`	Display the tool's version number.
`-w`	Toggle the "warnings" flag. When enabled, print out warning messages as the tool executes. The initial state of this flag is enabled.

The parameter `-p` can appear multiple times on the command line. For example,

```
   tool -p /h/tst -C cmds -p ../tmp -R responses -p /h/tst2 -C cmds
```

uses the command-line parameter `-p` to specify the location of three files: `/h/tst/cmds`, `/h/tmp/responses`, and `/h/tst2/cmds`. The default path for files when the parameter `-p` does not appear is tool-specific, but is usually `/h`.

Tool Helper Functions

The DII COE tools are designed to be extensible to accommodate new capabilities as new technologies are employed, such as CORBA or DCE. To provide this extensibility, the tools are designed to accept *helper functions*. Helper functions are “bolt ons” invoked by the tools whenever they encounter a segment descriptor that they are unable to process. For example, the database-related segment descriptors are processed by helper functions. When a tool, such as `VerifySeg`, encounters a database-related descriptor, it invokes the helper function to process (i.e., validate or otherwise respond to) the descriptor. The DCE, Web, and Database-related descriptors are processed by helper functions.

Helper functions require prior authorization from the DII COE Chief Engineer before they can be used with any of the tools in this appendix. Once authorization is granted, modification to the affected tools is required to recognize new helper functions. This is an intentional design restriction. Configuration of the tools must be carefully managed so that one helper function does not interfere with another and so that a helper function does not interfere with the consistent operation of the tools across all segment types.

Not all of the tools listed in this appendix accept helper functions. Helper functions are only useful when processing segment descriptors during the development process or during the installation process. Thus, tools such as `VerifySeg` and `COEInstaller` have helper functions, while tools such as `COEMsg` do not.

Summary Tables

Table C-1 and Table C-2 are an abbreviated list of the tools that indicates the following:

· which tools have a library interface as well as exist as an executable
· which tools accept helper functions
· which tools are available only on NT platforms
· which tools are available only on UNIX platforms.

If a tool is not listed, it means the following:

· the tool is available as an executable only; it does not have a library interface
· the tool does not accept a helper function
· the tools is available for both NT and UNIX platforms.

Tool	Library I/F Avail	Helper Fnct Avail	Supported Platforms
1. COEAddDBUser	Y		U
2. COEAskUser	Y		*
3. COEBackupDB	Y		U
4. COEConnectAS			U
5. COECreateAS			U
6. COECreateDS	Y		U
7. COEDeleteDBUser	Y		U
8. COEDropDS	Y		U
9. COEExtendDS	Y		U
10. COEFindServer	Y		U
11. COEInstaller		Y	
12. COEInstError	Y		*
13. COELstDBDepends			U
14. COELstDBRoleUsrs			U
15. COELstProfileSegs			U
16. COELstProfileUsrs			U
17. COELstSegProfiles			U
18. COELstSegUsrs			U
19. COELstUsrDBRoles			U
20. COELstUsrProfiles			U
21. COELstUsrSegs			U
22. COEMsg	Y		*
23. COEPrompt	Y		*
24. COEPromptPasswd	Y		*
25. COERebuildIndex	Y		U
26. COERebuildView	Y		U
27. COERemoveAS			U
28. COERestoreDB	Y		U
29. COEScanCOTS			N
30. COEStartDBServer	Y		U
31. COEStopDBServer	Y		U
32. COETestInstall		Y	*
33. COETestRemove		Y	*
34. COEUpdateHome	Y		U

Legend:				
	Y	- capability exists	**'blank'**	- no capability
	U	- UNIX only feature	**N**	- NT only feature
	*	- All platforms		

Table C-1: COE Runtime Tools

Tool	**Library I/F Avail**	**Helper Fnct Avail**	**Supported Platforms**
1. CanInstall		Y	*
2. ChkCompliance		Y	*
3. TestInstall		Y	*
4. TestRemove		Y	*
5. VerifyCOE		Y	*
6. VerifySeg		Y	*

Legend:				
	Y	- capability exists	**‘blank’**	- no capability
	U	- UNIX only feature	**N**	- NT only feature
	*	- All platforms		

Table C-2: COE Developer Tools

That is, the tables list the tools only on an exception basis. As the legend indicates, a “Y” in the Library I/F column indicates that a library interface exists for the tool while a blank indicates it does not. Similarly, a “Y” in the Helper Fnct column indicates that the tool accepts a helper function while a blank indicates that the tool does not. An “N” in the Supported Platform column indicates that the tool is available for NT platforms only, a “U” indicates it is available for UNIX platforms only, while an “*” indicates the capability is available for both platforms.

COE Runtime Tools

This section lists the COE tools that are available at run time. These executables are delivered to the operational site and are located underneath the directory `/h/COE/bin`.

COEAddDBUser

The creation and maintenance of COE database server user accounts are integrated with COE system management functions for creating and maintaining users' operating system accounts. Information on database users must be maintained so that they may be restored when a data store is restored. Restore files must be built by this application.

This function supports the discretionary granting of access within the database. It synchronizes users' permissions inside the database with their application profiles outside the database. It uses the database role information under the application segment's `Database` descriptor in `SegInfo` to do this.

This function is accessed automatically when accounts are added with profiles that include database access. It may also be accessed independently as part of the Database Administrator account group.

COEAskUser

This tool is intended for use in the `PostInstall` script to display a message to the user and have the user click on a "Yes" or "No" button. The syntax is

```
   COEAskUser {parameters} msg
```

where *msg* is the prompt to display to the user. The parameters allow the calling routine to specify the labels to be used for the "Yes/No" buttons.

COEBackupDB

The standard dump and restore format for the RDBMS will be provided. Special format data backup options will also be available as specified by the data store segments. This feature is available from the Database Administrator account group.

COEChkUpdates

The `COEChkUpdates` tool is invoked from the System Administration account group. It allows specification of a "master" machine to be checked for the availability of new segments, segment updates, or segment patches. The checking may be done on demand, or at a specified interval (e.g., every 24 hours). The default operation is to check on demand.

When `COEChkUpdates` determines that system updates are available, a notification message is displayed to the logged in operator that new features are available. A list is *not* displayed, only a notification is presented. The system administrator may then log in, invoke `COEChkUpdates` on demand, and receive a list of available new features for downloading.

COEConnectAS

`COEConnectAS` connects a client platform to an application sever. This tool is selected from the System Administration account group. The client platform must have at least the COE kernel installed, but may also have other segments loaded locally to improve performance. By default, `COEConnectAS` does not load segments from the application server onto the client platform. They remain on the application sever and are loaded into the client platform's memory as required for execution. However, `COEConnectAS` provides a feature that allows dynamic loading of segments. This feature allows segments to be loaded onto the client platform disk the first time an operator attempts to accesses a function in the segment. Future accesses are then done from the client platform's local copy. Dynamically loaded segments are removed from the client platform when the operator logs out, and a check is made each time a new operator logs in to a "dynamic" platform to be sure that segments were not inadvertently left on the client platform.

COECopyWS

`COECopyWS` is provided as an aid to speed up the site installation process. It allows segments on one platform, the source platform, to be copied across the network onto another platform, the target platform. This command will cause the target platform to be configured identically to the source platform, including ensuring that the COE kernel is installed and configured. Platform-specific items, such as the target platform's hostname and IP address are unmodified. Disk partitions and other actions normally performed when the COE kernel is loaded are not affected on the target platform.

The source platform must already have all segments installed that are to be copied to the target. The target platform must have at least the COE kernel installed, and must be reachable across the LAN from the source platform. The target platform must have sufficient disk space to hold all of the segments that are to be transferred. `COECopyWS` checks to be sure that this is the case, and reports an error message and aborts if it is not.

`COECopyWS` does not require that the target and source have identical disk partition configurations. However, segments that are grouped together on the same disk partition on the source platform will be grouped together on the same disk partition on the target platform.

This command is normally selected from the System Administration menu, but it may also be invoked from the command line. It may be executed only from the source platform.

COECreateAS

`COECreateAS` allows a site administrator to create an application server. It is normally selected from the System Administration account group and operates the same way as the `COEInstaller` tool. The difference is that this tool only loads the segments onto disk to be executed by another platform (see `COEConnectAS`), and this tool allows multiple hardware types to be loaded on the application server.

The `COECreateAS` tool does *not* support installation of multiple versions on the application server. This could lead to problems if two different versions of a segment, for the same platform type, are executed at the same time.

COECreateDS

`COECreateDS` creates data stores on a COE Database Server and allocates physical storage to the data stores. It also creates the database owner account associated with the data stores. `COECreateDS` is also used to create a new data store for an existing database owner. It must be executed when the database server is running. The purpose of `COECreateDS` is to isolate the database segment developer from the physical storage implementation of the various DII database server configurations.

Should `COECreateDS` encounter an error it cleans up any files, accounts, or other objects that were created during processing.

COEDeleteDBUser

This function supports the discretionary revocation of access within the database. It synchronizes users' permissions inside the database with their application profiles outside the database. It uses the database role information under the application segment's `Database` descriptor in `SegInfo` to do this.

This function is accessed automatically when accounts are deleted with profiles that include database access. It may also be accessed independently as part of the Database Administrator account group.

COEDelUnused

This tool examines all of the installed segments and determines which are unused (e.g., not in any profile, no other segment is dependent upon it). The list is presented to the system administrator who may then delete one or more of the segments. The purpose of this tool is to aid site administrators in finding and removing segments that are not be used in order to free up disk space.

This command is normally selected from the System Administration menu, but it may also be invoked from the command line. It may be executed from one platform to examine itself or another platform.

COEDropDS

`COEDropDS` removes the files associated with an existing data store from the database server. It is intended to be executed after all data objects have been removed using a database segment's `DEINSTALL` script. Accordingly, `COEDropDS` will fail if any data object exists in the data store being removed. `COEDropDS` cannot be used to remove components of the DBMS Server.

If neither a store name nor a store list is provided, `COEDropDS` will attempt to remove all data stores associated with the specified DBO and then to remove the DBO account from the DBMS. Should `COEDropDS` encounter an error it will restore any files, accounts, or other objects that were deleted during processing.

COEExtendDS

`COEExtendDS` adds additional physical storage to an existing data store. For `COEExtendDS` to be used the database owner account and the data store name must both exist. If the intent is to add a new data store (e.g. additional "Sybase segment") to an existing DBO, then `COECreateDS` should be used.

Should `COEExtendDS` encounter an error during processing it cleans up any files or other objects that were created before the error occurred. Also, the API will return an error status to segment installation script.

COEFindSeg

This tool returns information about a requested segment. The tool accepts as input the name of the directory where the segment is expected to be, the segment name, and the segment prefix. If the directory is omitted, or the expected directory is not there, the tool performs a search for the segment in the legal directory locations for segments identified in Chapter 5 (e.g., `/h`, `/h/AcctGrps`, `/h/COE/Comp`).

The tool returns the absolute path of the directory where the segment was found, the segment name found, the segment prefix, and the segment type, including the segment attribute if applicable.

COEFindServer

`COEFindServer` determines the hostname, IP address, and whether or not the server is a DCE server, for the requested server. The server name is the name specified by the `$SERVERS` keyword in the `Network` descriptor, or in the `DCEServerDef` descriptor. If the server is a DCE server, the hostname and IP address returned will be of the first host where the server is found. There may be multiple hosts running the requested server.

COEInstaller

`COEInstaller` is normally executed from a System Administrator menu by an operator who has system administration privileges. It displays a list of configuration definitions or segments that may be installed from tape, disk (e.g., a network segment server), or other electronic media. It may also be executed directly from the command line.

By default, `COEInstaller` does not write any output to `stdout`. The verbose and warning flags are both disabled unless enabled by the `-v` and `-w` parameters respectively.

The `COEInstaller` writes information to a status log that indicates installation progress, which segments have been installed, and other information that might be useful to the site administrator. The site administrator may also choose to have the `COEInstaller` log information that is generated from the `PreInstall`, `PostInstall`, and `DEINSTALL` scripts. That is, the output from these scripts is "piped" to the status log. This capability is useful in the event problems are encountered during the installation process. The `COEInstaller` also provides an option to print the status log.

COEInstError

`COEInstError` allows a segment to display an error to the user from within a `PreInstall`, `PostInstall`, or `DEINSTALL` script and terminate installation of the segment. `COEInstError` *always* returns the same return code and will cause `COEInstaller`, `TestInstall`, and `TestRemove` to halt further processing of the segment.

If the calling descriptor (e.g., `PreInstall`, `PostInstall`, or `DEINSTALL`) does execute `COEInstError`, `COEInstaller` will detect that `COEInstError` has been called, terminates segment installation, and will display an appropriate error message. Whether or not the calling script sets a "terminate" return code, the `COEInstaller` will detect that `COEInstError` has been called and terminate segment installation.

`COEInstError` must be used carefully. If it is invoked from within `PreInstall` or `PostInstall`, the installation tools will remove the segment because the installation has been unsuccessful. If invoked from within `DEINSTALL`, the installation tools will *not* remove the segment.

COEListDepends

`COEListDepends` is normally selected from a System Administration menu. It may also be executed from a command line. It displays a sorted list of all segments upon which a specified segment depends. If no segment name is specified, a GUI is displayed and the user is prompted to select a segment from a list of segments accessible on the machine. If a segment name is specified, it is assumed that the tool is being run from a command line and the tool writes the output to `stdout` only instead of a window.

COEListSegs

`COEListSegs` is normally selected from a System Administration menu. It displays a sorted list of all segments that have been installed on a specified machine. The tool may be run in command-line mode only (with output written to `stdout`), or through a GUI.

COELstDBDepends

`COELstDBDepends` executes within the DBMS to search object dependencies across multiple database segments. Depending on the input parameters, it will search for all dependencies of a specified data object or for all dependencies of a database owner's schema on other schema's objects.

COELstDBRoleUsrs

`COELstDBRoleUsrs` provides a sorted list of all user's database login accounts assigned to a specific database role. This function is available from the Database Administrator account group.

COELstProfileSegs

`COELstProfileSegs` is normally selected from a System Administration menu. It may also be executed from a command line. It displays a sorted list of all segments accessible from a specified profile. If no profile is specified, a GUI is presented from which the user can select a profile. If a profile is specified, it is assumed that the tool is being run from a command line and all output is sent to `stdout`.

COELstProfileUsrs

`COELstProfileUsrs` is normally selected from a System Administration menu. It may also be executed from a command line. It displays a sorted list of all user login accounts (designated as local or global) that are assigned to a specific profile, where the profile may be global or local.

If no profile is specified, the user is prompted to select a profile from a list of profiles available on the machine. If a profile is specified, it is assumed that the tool is being run from a command line and the tool writes the output to `stdout` only instead of a window.

COELstSegProfiles

`COELstSegProfiles` is normally selected from a System Administration menu. It may also be executed from a command line. It displays a list of all profiles sorted by account group that contain a specified segment.

If no segment name is specified, the user is prompted to select a segment from a list of segments accessible from the machine. If a segment name is specified, it is assumed that the tool is being run from a command line and the tool writes the output to `stdout` only instead of a window.

COELstSegUsrs

`COELstSegUsrs` is normally selected from a System Administration menu. It may also be executed from a command line. It displays a sorted list of all user login accounts (designated as local or global) that have access to a given segment.

If no segment name is specified, the user is prompted to select a segment from a list of segments accessible from the machine. If a segment name is specified, it is assumed that the tool is being run from a command line and the tool writes the output to `stdout` only instead of a window.

COELstUsrDBRoles

`COELstUsrDBRoles` provides a sorted list of all database roles assigned to a specific user. This function is available from the Database Administrator account group.

COELstUsrProfiles

`COELstUsrProfiles` is normally selected from a System Administration menu. It may also be executed from a command line. It displays a list of all profiles sorted by account group that are assigned to a specific user.

If no user login account name is specified, the user is prompted to select a login account from a list of local and global accounts accessible from the machine. If a user login account name is specified, it is assumed that the tool is being run from a command line and the tool writes the output to `stdout` only instead of a window.

COELstUsrSegs

`COELstUsrSegs` is normally selected from a System Administration menu. It may also be executed from a command line. It displays a sorted list of all segments that are accessible by a specific user.

If no login account name is specified, the user is prompted to select a login account from a list of local and global accounts accessible from the machine. If a login account name is specified, it is assumed that the tool is being run from a command line and the tool writes the output to `stdout` only instead of a window.

COEMsg

This tool is intended to be used by `PreInstall`, `PostInstall`, and `DEINSTALL` to display an informational message to the user. A message is displayed in a window and the user must click on an "OK" button to continue.

COEPrompt

This tool is similar to `COEMsg`, but expects the user to enter a response. The calling routine may indicate the maximum number of characters in the user's typed response. The default, if not specified, is 40 characters.

COEPromptPasswd

`COEPromptPasswd` is similar to `COEPrompt` in syntax and operation. It is intended to be used in `PreInstall` and `PostInstall` to prompt a user to enter a password. This routine displays a window in which the user may enter a password. The user's response is not echoed in the window.

By default, `COEPromptPasswd` will prompt for the password, then prompt the user to enter the password again for confirmation. A flag may be passed to the tool to indicate that a confirmation prompt is not desired. The default maximum password length that a user can enter is 14 characters, while the minimum is 6 characters. As with `COEPrompt`, a parameter to this tool allows the minimum and maximum to be set to other values.

When confirmation is requested, the tool prompts the user for a matching confirmation up to 3 times before returning a failure. If after 3 attempts the confirmation does not match, the tool returns an error code and it is up to the calling routine to determine what action to take. This works the same way whether the tool is invoked via a library interface, or as a separate executable program.

COEPropagateUpdates

This tool is available only to the system administrator. It allows segments (new segments, upgrades, and patches) to be sent to other platforms. It is used in conjunction with `COEChkUpdates` to automatically update platforms on the LAN as new upgrades are available.

One possible scenario is to use `COEChkUpdates` to determine that new segments are available. Then, the segments are downloaded to a network installation server and temporarily installed (`COETestInstall`) for testing. Once verified as acceptable at the site, `COEPropagateUpdates` can be used to determine which platforms on the LAN need to be upgraded and then automatically perform the upgrades.

COERebuildIndex

`COERebuildIndex` is used to recover or restore corrupted indexes within the database server. It recreates the specified index and is available from the Database Administrator account group.

COERebuildView

`COERebuildView` is used to recover or restore corrupted views within the database server. It saves any existing grants to users or database roles and then recreates the specified view. It then re-executes the saved grants to finish restoring the view. This function is available from the Database Administrator account group.

COERemoveAS

This tool removes a segment from an application server (see `COECreateAS`). `COERemoveAS` is selected from the System Administration account group. When all segments in a particular hardware type are removed, all directories associated with the hardware type on the application server are removed.

COERestoreDB

The restore will not only restore the backup data store (data and structure), it will also run the Restore scripts for other data store segments to complete the restore process. The owning segment will tag the Restore script with “.main”. This script will be run first, followed by other public data stores, and then private data stores which have Restore scripts for the data store being restored.

This feature is available from the Database Administrator account group.

COERmtInstall

`COERmtInstall` is the remote installation tool. It is normally selected from the System Administration menu, but may also be invoked from the command line. The tool operates in either a "push" or a "pull" mode.

Push Mode

Push mode operation consists of an operator at a repository site (e.g., DISA Operational Support Facility) sending one or more segments from the repository site to the remote site. The “pushed” segment is sent to a remote network’s installation server and when the transfer is completed, the tool can be asked to automatically invoke the `COEInstaller` tool on the remote machine to allow the operator from the sending site to perform an actual installation.

Pull Mode

Pull mode operation consists of an operator at a remote site requesting the transfer of one or more segment install file(s) from the repository site to the remote site. In pull mode, the "pulled" segment is loaded onto the remote site's installation server from which the operator may install the segment on one or more machines.

Remote Segment Removal

`COERmtInstall` also allows a repository site to remotely remove a segment on a machine at the remote site. This is essentially accomplished by launching `COEInstaller` on the remote machine and using it to select the segment(s) to delete.

`COERmtInstall` provides several security measures (encryption, passwords, anonymous ftp, etc.) to protect segment transmission and to prevent unauthorized access to the repository or a remote site. Discussion of such measures is beyond the scope of this document.

COEScanCOTS

The `COEScanCOTS` tool is available only on NT platforms. It scans the system to find applications that are already installed and automatically creates segment descriptors for them. This allows the COE to be loaded on an already established NT environment where several COTS products may already have been loaded. Then, with segment descriptors created for these pre-loaded COTS products, segments loaded on later can express dependencies on them.

COEStartDBServer

`COEStartDBServer` will check to see if the specified server is running and if not, it will start the DBMS to support the installation of the a database segment.

Note: This tool starts the database server in its maintenance mode.

COEStopDBServer

`COEStopDBServer` supports the shutdown of the DBMS as necessary to support the installation of a data base segment. The DBMS server will be located by the supplied name using the interfaces and hosts file and stopped using the DBMS commands for a normal shutdown of the server.

COETestInstall

The `COETestInstall` operates exactly like the normal `COEInstaller`, except that its purpose is to temporarily install a segment on a platform. It is normally selected from the System Administration account group, but may be executed from a command line. The temporary installation may be an individual segment, or a collection of segments from a configuration definition. The purpose of a temporary install is to allow a site administrator to temporarily install a segment on an isolated test platform for the purposes of checking out the segment before committing it to widespread installation.

When a segment is temporarily installed, an expiration date may be entered. The default is 48 hours. When the expiration date arrives, a warning message is displayed to the operator upon login to indicate that a segment has expired. However, the segment is *not* removed and the notification is for warning purposes only. `COETestRemove` must be invoked to actually remove the segment. If the test is successful, the `COETestInstall` tool can also be used to “promote” the segment from test status to “installed” without the need to delete and reinstall the segment.

COETestRemove

`COETestRemove` is selected from the System Administration account group and is used to remove a segment that was temporarily installed. Its interface looks the same as the `COEInstaller`, but it only lists temporarily installed segments and will not allow the removal of any other type of segment.

Removal of a temporarily installed segment can be done at any time, whether or not the expiration has occurred. The operator has the option of either removing the segment, or making the installation “permanent.” When a temporary segment is removed, the segment it replaced, if any, is restored.

COEUpdateHome

`COEUpdateHome` is intended to be invoked from within a segment's `PostInstall` script. Its purpose is to update the home environment variable within a script file to point to where a segment was actually installed.

This tool searches the named script file for the first place that the environment variable is defined through either `set` or `setenv`, and replaces the definition with the absolute pathname for where the segment was loaded.

For example, assume `COEInstaller` loads `MySeg` underneath `/home4/MySeg`, and the script file `.cshrc.MYSEG` contains the following:

```
setenv MYS_HOME        /h/MySeg
```

Then the statement

```
COEUpdateHome Scripts/cshrc.MYSEG MYS_HOME
```

changes the statement to read

```
setenv MYS_HOME        /home4/MySeg
```

Note: Since environment extension files are not required in the NT COE, `COEUpdateHome` is not available for NT platforms.

COE Developer Tools

This section lists the COE tools that are available during development, but are not delivered to operational sites. By default, these executables are located underneath the directory `/h/TOOLS/bin` and are distributed as part of the Developer's Toolkit. These tools are not location sensitive, and may be moved to any directory desired for development.

CalcSpace

`CalcSpace` computes the space required for the segment specified and updates the `Hardware` segment descriptor accordingly. The segment referenced must *not* be compressed, and must not contain any files that do not belong with the segment (e.g., source code) at runtime. Otherwise, the space computed will be incorrect.

The `-v` flag enables the verbose flag and causes the tool to print out the space requirements for each top-level subdirectory. The warning flag is enabled by default and causes the tool to print a warning message if directories are found that are not expected (e.g., `include`, `src`) or if expected directories are missing for the segment type (e.g., `bin` for software segments). `CalcSpace` does not affect the reserve space request in the `Hardware` descriptor, and does not affect any partitions specified.

The amount of space required is reported in K bytes.

CanInstall

`CanInstall` tests a segment to see if it can be installed. It performs the same tests that `COEInstaller` does at installation time. To be installable, all required segments must already be on the disk, and there must not be any conflicting segments on the disk.

ChkCompliance

The `ChkCompliance` tool examines a segment to determine its Category 1 compliance level. Some requirements cannot be checked automatically (e.g., using only POSIX calls, expected location of X/Motif libraries), so the tool reports the *maximum* possible level of compliance based on the criteria that can be checked automatically.

ChkDBCompliance

`ChkDBCompliance` examines a database segment’s database to determine its level of compliance. It executes within the DBMS to complement `ChkCompliance`. It cannot be executed independently, because it is a helper function for `ChkCompliance` and is called automatically from `ChkCompliance`.

CMMakeDistrib

This tool is intended for use by Configuration Management departments. It is a configurable tool that allows groups to be defined (e.g., operational sites, developers) and to package all or portions of a submitted segment for delivery to the selected group.

For example, an operational site does not need libraries and header files but developers do. Developers require these to be able to create application segments, but source code is not normally sent to developers for packages written by another developer. For the operational site, `CMMakeDistrib` would ensure that header files and libraries are excluded from the delivery, but are present for the package sent to a developer. `CMMakeDistrib` extracts the necessary directories out of the repository and provides them to `MakeInstall` for creating the actual distribution media.

COEIDE

The `COEIDE` (COE Integrated Development Environment) tool provides a GUI for accessing all of the developer tools in subsection C-0, and any runtime tools (such as `COEInstaller`) that are useful during the development process. The tools accessible from `COEIDE` are limited to those which operate on segments, such as `VerifySeg` or `MakeInstall`, but not runtime tools such as `COEMsg` that are invoked by the segment itself.

`COEIDE` is a menu-and-icon-driven interface that allows the developer to open segments for development work. Once opened, any of the development tools can be run against the segment. `COEIDE` allows the output from tools to be captured into a log file for later printing or review. It also includes the ability to set tool flags such as the verbose and warning flags.

The purpose of `COEIDE` is to provide a simplified, single interface to all of the tools that are useful to create segments or retrieve them from a repository, validate them, test them, and then submit them electronically to the appropriate SSA. In this case, the SSA could be within the developer's own organization so that the tool can be used to transition a segment from the developer to configuration management, and to integration and test organizations.

`COEIDE` is available for both the UNIX and the NT environments.

ConfigDef

`ConfigDef` is used to create a configuration definition from a list of segments. Components of a configuration definition (e.g., bundles, configurations, folders) may participate in more than one configuration definition.

Segments will normally reside in a segment repository such as SDMS. Thus, `ConfigDef` provides an interface that allows segments to be checked out of the repository. The interface is the same as that provided by `MakeInstall`.

ConvertSeg

`ConvertSeg` is a tool that examines segment descriptors and converts them to the latest format. To the extent possible, obsolete usage is identified and corrected as part of this process. The original segment descriptor directory is not modified, but it is renamed to be `SegDescrip.orig`. The converted segment descriptors created by the tool are placed in a newly created (e.g., after the original `SegDescrip` is moved to `SegDescrip.orig`) `SegDescrip` directory. To avoid inadvertently overwriting a segment descriptor directory, a prompt is issued if the directory `SegDescrip.orig` already exists.

This command is useful in combining individual segment descriptor files into the proper `SegInfo` descriptor. The tool will print a warning if obsolete directories, such as `progs` and `libs`, are encountered.

MakeAttribs

This tool creates the descriptor `FileAttribs` described in Chapter 5. It recursively traverses every subdirectory beneath the segment home directory and creates a file with lines in the format

```
permits:owner:group:filename
```

At installation time, the installation tools perform the following statements for each entry:

```
chmod permits $INSTALL_DIR/filename
chown owner $INSTALL_DIR/filename
chgrp group $INSTALL_DIR/filename
```

For security reasons, `MakeAttribs` does *not* include any file owned by root nor any file for which *permits* is greater than 777. A warning is printed for each filename that is rejected. A warning is also printed for each "suspicious" permission setting such as 777 for a file, execute permissions on files in a `data` subdirectory, etc.

MakeInstall

The `MakeInstall` tool writes one or more segments to an installation medium, or packages the segments for distribution over SIPRNET. `MakeInstall` checks to see if `VerifySeg` has been run successfully on each of the segments and aborts with an error if it has not. `MakeInstall` supports multi-volume tapes, but will not split segments across tapes. It allows segments for different hardware platforms to be on the same installation medium and it also allows configuration definitions to be written to the installation medium.

Command-line parameters and segment lists may be repeated as required. A command-line parameter applies to all succeeding command-line arguments until the next parameter is encountered.

All components of an aggregate must be included at the same time. The order in which segments are passed to `MakeInstall` is unimportant because the tool will order them in such as way as to guarantee that a tape would not need to be rewound during installation.

`MakeInstall` requires that segments be available on disk with at least the `SegDescrip` subdirectories in an uncompressed, unencrypted format. Since segments are normally installed in a repository such as SDMS, `MakeInstall` provides an interface to allow segments to be checked out from the repository as required. This is done by invoking a user-supplied program and passing it the name of the segment to extract. The supplied program checks the requested segment out of the repository and places it in an agreed-upon temporary location. `MakeInstall` automatically deletes the temporary segments when finished with them.

mkSubmitTar

Segments must be packaged by compressing and encrypting them prior to submitting them electronically to SDMS. `mkSubmitTar` does this task. It checks to see if `VerifySeg` has been run successfully on the segment and aborts with an error if it has not. `mkSubmitTar` also ensures that the directories specified in Chapter 5 as required for segment submission are provided.

`mkSubmitTar` allows segments to be checked out of a repository prior to packaging for submission. `mkSubmitTar` will only package one segment into a tar file. Each component of an aggregate is packaged separately. However, `submit` allows multiple files to be sent during one session. An entire aggregate should be sent at one time unless the size is prohibitive.

MoveSeg

`MoveSeg` allows a segment that has already been installed to be moved from one location to another, including to another disk partition. An error will be generated if there is not enough room to move the segment. The old segment location is deleted when the move is completed.

SegAnalyze

`SegAnalyze` analyzes the segment specified and creates a table showing total memory and disk usage. It determines all segment dependencies so that the statistics reported include all required segments.

Note: `SegAnalyze` was called `VarAnalyze` in the previous *I&RTS*.

SegCatalog

This tool adds a segment to the segment catalog.

submit

This tool electronically transmits segments packaged by `mkSubmitTar` to the host repository. By default, segments are *not* deleted from the submitting machine after transmission.

TestInstall

`TestInstall` is used to temporarily install a segment that already resides on disk. It must be run when there are no other COE processes running. The reason for this restriction is that the tool may modify COE files already in use with unpredictable results. `VerifySeg` must have been run before `TestInstall` to make sure that the segment is valid.

`TestInstall` performs the same operations as `COEInstaller` except that it does not need to read the segment from tape (e.g., it is already on disk), and the segment may be in any arbitrary location. `TestInstall` will establish the required symbolic link under `/h` to preserve the COE standard directory structure.

Note: `TestRemove` should be used to remove a segment installed by `TestInstall`. `COEInstaller` should *not* be used because it will physically delete the segment from disk while `TestRemove` will not.

TestRemove

`TestRemove` is used to remove a segment that was installed by `TestInstall`. It must be run when there are no other COE processes running. The reason for this restriction is that the tool may modify COE files already in use with unpredictable results. `TestRemove` removes the symbolic link under `/h` if one exists, but it does *not* delete the segment from disk.

Note: `TestRemove` is unconditional and should be used with great caution. It will remove the effects of installing a specified segment even if other segments still installed depend upon it. This tool does *not* delete the segment from disk.

TimeStamp

`TimeStamp` puts the current time and date into the `VERSION` segment descriptor. It is intended to assist the configuration management process by allowing the time stamp to be updated just prior to running `VerifySeg` and `mkSubmitTar` for the deliverable product.

unpackSubmitTar

This tool will decompress and decrypt segments that have been packaged with the `mkSubmitTar` tool.

VerifyCOE

The tool `VerifyCOE` is used to validate the COE runtime environment. It is intended to be used after the COE kernel has been loaded and configured to ensure that it has been correctly set up. This tool is primarily intended for developers who wish to alter the COE installation instructions to conform to development environment needs.

Specific tests performed include:

- Check for sufficient swap space on the disk.
- Check for correct disk partitioning.
- Check for expected device drivers.
- Check for sufficient shared memory, message pool size, and other operating system kernel parameters.
- Check for proper location of X and Motif libraries.

VerifySecurity

The `VerifySecurity` tool is used to perform security checks against a segment (similar to the operation of `VerifySeg`), against an installed COE (similar to `VerifyCOE`), or against an installed system. A single security policy cannot be stated for all COE-based systems, so this tool is limited to identifying potential security risks that are common across all systems (e.g., files that are world readable/writeable, use of remote commands, improper .rhosts files, etc.). When used against an entire system, the tool interfaces to commercially available tools such as `Satan` to identify vulnerability areas. Because of the serious potential for virus attacks, especially in the NT world, the tool also performs virus checking through an interface to commercially available virus checkers.

`VerifySecurity` checks security-related issues to the extent possible. Neither it nor any other tool is fully comprehensive, nor does it obviate the need for a disciplined security policy and security enforcement procedures.

VerifySeg

`VerifySeg` is used to validate that a segment conforms to the rules for defining a segment. It uses information in the `SegDescrip` subdirectory and must be run whenever the segment is modified. `VerifySeg` must be run for each segment. If the segment is an aggregate segment, `VerifySeg` must be run on each segment in the aggregate.

Specific tests include:

· Check to ensure that executables are prefaced with the segment prefix where required.
· Check to ensure that all defined environment variables use the segment prefix.
· Check to ensure that all pathnames are defined relative to the home environment variable, `segprefix_HOME.`
· Check to ensure that all required environment variables (`USER_HOME`, `USER_DATA`, `DATA_DIR`, etc.) are defined by account group segments.
· Check to ensure that all required environment variables are defined by the COE parent component segment.
· Check aggregate segments for internal consistency (e.g., all components are hardware compatible).
· Check and warn if `mv` is used in a `PostInstall`, `PreInstall`, or `DEINSTALL` script since this may be an illegal attempt to move files across disk partitions.

VerifySegDB

`VerifySegDB` is used to validate that a database segment's database conforms to the rules for defining a segment's database. It executes within the DBMS and is a helper function for `VerifySeg`. It cannot be executed independently of `VerifySeg`, but is called automatically from `VerifySeg`.

VerUpdate

`VerUpdate` updates the segment version number, date, and time in the `VERSION` descriptor. Command-line parameters can be used to indicate which digits (e.g., major release, minor release, maintenance digit, or developer digit) to update. Multiple digits may be updated at one time. By default, only the maintenance digit is updated.

If no version number is specified, the version number inside the `VERSION` descriptor is incremented. That is, 1.0.0.0 is updated to be 1.0.0.1. If no version is specified, and `VERSION` does not exist or has no version number, the file is created and version number 1.0.0.0 is inserted.

Note: No error checking is performed on the version number specified since the segment must still be validated by `VerifySeg`.

This page is intentionally blank.

Appendix D: Accessing COE Online Services

This appendix describes how to access the COE online services described in Chapter 10. These services are used to view documentation, the segment catalog, download files, and participate in project-related newsgroups.

The primary method for accessing online services is via the World-Wide-Web. The World-Wide-Web is a virtual network existing on top of the Internet. It is based on a hyperlink protocol called HTTP (HyperText Transfer Protocol). Users access the web through any available hypertext browser, such as:

- Mosaic, developed by the National Center for Supercomputing Applications
- Netscape, marketed by Netscape Corporation
- MSN, the Microsoft Network.

Web servers are represented by Uniform Resource Locators (URLs). The URL for the unclassified (Internet) DII COE online services is

```
http://spider.osfl.disa.mil/dii
```

Note: The lowercase letter "`l`" is used in "`osfl`" and not the number "`1`."

This URL contains useful COE information, and contains hyperlinks to a number of related government home pages. Specific information available from the DII COE home page includes:

- Instructions for downloading COE-related documents

- A search engine for making queries
- SHADE Documentation and news
- Working papers from the COE Engineering group
- Meeting minutes from the DII COE Architectural Oversight Group
- Meeting minutes from the various Technical Working Groups
- Schedule of upcoming events, including product releases
- Form for submitting requests to the DISA DII CM department
- Frequently Asked Questions

Other useful links include:

`http://www.disa.mil`	Home page for DISA
`http://www.disa.mil/line/jieo.html`	Home page for DISA JIEO
`http://spider.osfl.disa.mil`	Home page for GCCS-related information

Meeting notices and minutes, briefings, and documentation can be retrieved and downloaded from the DISA web server. Most of the information can be viewed directly online through a web browser.

All information at the given URL is unclassified. Contact the DISA Engineering office to receive the URL for the classified (SIPRNET) COE online services. These URLs are subject to change as the COE is extended.

The remainder of this appendix is being revised as the COE services are extended to support GCSS, GCSS, ECPN, and other COE-based systems. An update to this appendix will be provided when completed. Specific information will be provided then for:

- accessing services via anonymous ftp
- viewing the segment catalog
- downloading segments from the SDMS
- downloading the Developer's Toolkit
- downloading documentation
- searching for specific documents or topics
- accessing COE-related newsgroups
- receiving event notifications related to segments in progress
- retrieving meeting minutes, briefings, and meeting notices.

Appendix E: Registering Segments

Segment registration, whether COE-component segments or mission-application segments, is required for all software and data submitted to DISA[11]. Segment registration

[11] All COE-component segments must be registered with the DISA SSA. All mission-application segments for DISA systems (e.g., GCCS, GCSS, ECPN) must be registered with the DISA SSA. Other

is used to collect information about a segment so that it can be published in the Segment Catalog, and so that a segment prefix and segment directory can be assigned.

As a minimum, segment registration requires the following information about each segment:

Segment Information

· Segment Name (full descriptive title)
· Target System (GCCS, GCSS, COE component, etc.)
· Segment Type (data, software, account group, etc.)
· Segment Prefix
· Segment Home Directory
· Resources Required (memory and disk estimates, requested port assignments, reserved user IDs, etc.)
· List of boot and background processes
· Platform Availability (PC only, Solaris only, etc.)
· Programming Language(s)
· List of Related Segments (if applicable)
  - Aggregate Components
  - Required Segments (including version and patch requirements)
· Operating System(s) Supported
· Windowing Software Required (including version dependencies)
· Other Required COTS Products (including version dependencies)
· Available Documentation (User's Guides, Programming Guides, etc.)
· For database segments, both the usage and source of the data

Management Information

· Name, Organization, Address, Phone, Fax, and email Address for Primary and Optional Alternate Program Management Point of Contact
· Name, Organization, Address, Phone, Fax, and email Address for Primary and Optional Alternate Technical Point of Contact
· Name, Organization, Address, Phone, Fax, and email Address for Primary and Optional Alternate Process Point of Contact
· Special Restrictions (e.g., not releasable, classified)
· Expected segment release date

---

mission-application developers should contact the cognizant DOD program manager to receive registration instructions and determination of the applicable SSA.

Miscellaneous Catalog Information

- Short paragraph describing segment features and purpose
- List of keywords for use in catalog searches
- Unclassified picture of the segment user interface (GIF, JPEG, or X11 Bitmap format) (optional).

The information listed is required for each component of an aggregate. Notification of key events about the segment will be sent to the primary and alternate Process Point of Contact. The Process Point of Contact is responsible for disseminating the information to any other authorized individuals.

Once a registration form is received, DISA will contact the Program Management Point of Contact to verify that the submission is authorized. The segment prefix, segment name, and segment directory will be accepted as requested unless they conflict with another segment that is already registered. Where there is a conflict, DISA will assign a new prefix, name, or directory as required and notify the Process Point of Contact. DISA may also provide other items, such as authorization keys, when the segment has been registered.

Segments may be registered via a web browser or email. The information collected at segment registration time may be enhanced over time to support emerging program requirements. Contact the DISA Engineering Office for the proper email address and URL, for registration information updates, and for special handling (e.g., classified data) requirements.

Appendix F: Vendor-Specific DBMS Considerations

This appendix contains information that is DBMS vendor-specific. It will be updated as COTS versions change, or as other vendor products are supported.

Oracle RDBMS

This section contains information specific to the Oracle Database Server Segment and to database segments written for the Oracle RDBMS. It also provides information for developers of application segments that will access Oracle databases.

Oracle Server Segment

The Oracle RDBMS Server segment contains the software required to install and maintain an instance and a database, and the software to support both client/server and server/server communications across the network. It provides the executables for the RDBMS, and the SQL*Net network. It also contains the core database instance: Database files, tablespaces, and internal Oracle data structures (data dictionary, etc.) that are used in common by all databases within the DBMS. Finally, this segment has the tools used by database administrators to administer the DBMS and databases. Table F-1 shows the server configuration.

DBMS Executables	Oracle RDBMS, SQLDBA, Names Server, Server Manager, Export, Import, SQL*Plus, SQL*Loader
Application executables	SQL*Forms 3.0, Oracle*Forms 4.5, Oracle*Reports 2.5
Core database configuration	Tablespaces for System, Rollback Segments, Temporary, Tools and Users Log file archiving turned off
Data dictionary	Core data dictionary objects Support objects for performance monitoring Support objects for stored packages
Application database objects	Data objects for each executable Stored packages and procedures for each executable
DBA account	Operating System ‘`dba`’ account group ‘`dbadmin`’ account in ‘`dba`’ group DB Admin. only No ownership or direct connection

Table F-1: Oracle RDBMS Server Configuration

DISA builds the Oracle DBMS server segment in the following manner. Oracle's installer is used to load executables. Then the core database is built based on the aggregate sizing information provided by developers. Once the database is created, the data dictionary is loaded. The data dictionary load includes the core objects required for the DBMS to function, additional items to ease DBA tasks, and packages and procedures that allow developers to use the features of Oracle 7.

The core database is installed with two accounts, `sys` (the owner of the core database) and `system` (the database administrator). Security measures mandate that the passwords of both the sys and system accounts be changed upon successful installation of this segment. The use of scripts or applications containing hard coded passwords for these accounts, or any other Oracle database accounts, is prohibited.

Log file archiving is turned off in the initial DBMS configuration. This allows applications' database segments to be installed and their data loaded without generating massive numbers of useless log files. Once a site's configuration is completely loaded, the DBA at that site must turn archive logging on to support DBMS recovery.

The Oracle RDBMS provides centralized recovery features (Rollback Segments, Online Log Files, and archived Log Files). Their configuration is set when the COTS DBMS Segment is built by DISA. Developers who need specific configurations of these components must include their requirements in their Segment Registration. Any such modifications to the COTS DBMS Segment are subject to the review and approval of the cognizant DOD Chief Engineer and will be implemented by DISA or the cognizant DOD program office, not the developers.

The last component of the core database segment is a model database administrator account named '`oradba`.' This is a UNIX account that has the privileges needed for database administration. The '`oracle`' account that owns the executables is a special Solaris account without login capability. No user can connect to '`oracle`' directly. A user must login as a normal user, use the '`su`' command to become '`root`' and then connect to the '`oracle`' account.

Oracle Client Segment

The Oracle COTS RDBMS Client segment contains the Oracle RDBMS executables for to support client/server communications across the network. Depending on the needs of other segments it may also provide executables for accessing the database, tools and utilities to support data extraction, loading, and backups. The Oracle client services segment is loaded on each application server and on client platforms. Small client platforms may NFS mount the disk/directory containing the Oracle segment. It contains the runtime applications provided by Oracle (e.g., SQL*Forms 'runform') and the client portion of SQL*Net. DISA supplies different client segments for the various DII platforms to support their varying display and keyboard configurations. Table F-2 shows the client configuration.

Executables	SQL*Net (client), Oracle Forms (runform), Oracle Reports (runrep), Export/Import, SQL*Loader, SQL*Plus

Table F-2: Oracle RDBMS Client Configuration

Oracle Application Database Segments

There are some special considerations for mission applications that use Oracle services.

DBS_files Directory

The functional storage allocation of a Data Store is implemented in Oracle from within the DBMS as ‘tablespaces.’ Database segments using Oracle must create one or more tablespaces to store their objects (one tablespace for each Data Store). Segments are prohibited from storing tables and indexes in another segment’s tablespace, or in the SYSTEM, TEMP, USERS, or TOOLS tablespaces. A segment may create a tablespace composed of multiple files if the storage requirements are large enough to span physical devices.

All data files for the database segment will be place in the `DBS_files` directory. The `DBS_files` directory in an Oracle database segment must be owned by ‘`oracle`’ in group ‘`dba`.’ The change of ownership must occur before the CREATE TABLESPACE command is issued. The DBA must issue the CREATE TABLESPACE command.

Oracle Tables

The creation of Oracle tables must specify the storage parameters for controlling their size. In addition, the create script must explicitly specify the Oracle tablespace that is to store the table. The following script excerpt creates a database table in Oracle.

```
CREATE TABLE SORTSM_BIDES (
      UIC      VARCHAR2 (6)   NOT NULL
     ,SECUR    VARCHAR2 (2)   NOT NULL
     ,TIME     DATE      NOT NULL
     ,SERV     VARCHAR2 (1)   NULL
     ,ANAME    VARCHAR2 (30)  NULL
     ,UTC      VARCHAR2 (5)   NULL
     ,ULC      VARCHAR2 (3)   NULL
     ,SCLAS    VARCHAR2 (2)   NOT NULL
     ,MJCOM    VARCHAR2 (6)   NULL
     ,MONOR    VARCHAR2 (6)   NULL
     ,MAJOR    VARCHAR2 (1)   NULL
     ,REVAL    VARCHAR2 (1)   NULL
     ,UDC      VARCHAR2 (1)   NULL
     ,LNAME    VARCHAR2 (55)  NULL
     ,COAFF    VARCHAR2 (2)   NULL
```

```
          ,BUPDATE  DATE NULL
          ,TPSN     VARCHAR2 (7)   NULL)
     TABLESPACE DBSORT_DATA,
     STORAGE(
          INITIAL   2000000
          NEXT      500000
          MAXEXTENTS     99
          PCTINCREASE    0);
```

Oracle Views

Views should be created after all tables have been created to guarantee that the referenced table objects exist. A segment may create views to support legacy structures as well as views to support application-specific database requirements. The following example script excerpt creates a legacy view in Oracle to support a change in a column name from SECURITY to SECUR.

```
     CREATE VIEW SORTSM_BIDES_1_1
           (UIC, SECURITY, TIME, SERV, ANAME, UTC , ULC,
     SCLAS , MJCOM ,
            MONOR, MAJOR, REVAL , UDC, LNAME , COAFF,
     BUPDATE, TPSN)
      AS SELECT UIC, SECUR, TIME, SERV, ANAME, UTC , ULC,
     SCLAS , MJCOM,
            MONOR, MAJOR, REVAL , UDC, LNAME , COAFF,
     BUPDATE, TPSN
      FROM SORTSM_BIDES;
```

Oracle Constraints

When the install process includes a base load of data it can be done faster if the primary keys and indexes are created after the data is loaded. The following script excerpt creates a primary key constraint in Oracle.

```
     ALTER TABLE SORTSM_BIDES ADD PRIMARY KEY (UIC)
     USING INDEX TABLESPACE DBSORT_INDEX
     STORAGE (INITIAL 250K NEXT 100K PCTINCREASE 0);
     eof
     ;;
```

Note: `PCTINCREASE` should always be 0.

Oracle Packages

A database package is an Oracle feature that provides a way of grouping and storing related procedures, functions, cursors and variables together as a unit in the database for continued use. Packaged procedures and functions can be called explicitly by applications or users.

A package is dependent on all objects referenced in the constructs of its body. A package is successfully compiled only when all referenced objects are present and of the expected structure, and if the owner of the package has the necessary privileges for the referenced objects. A package becomes invalid if any of the objects referenced within the package is altered or dropped. When an application invokes a package’s public procedure or function, this application must have the privilege to execute the package. Having this privilege also allows the application to execute any construct within that package.

Oracle Indexes

An Oracle index should specify a tablespace and the indexes' storage parameters. The following script excerpt creates an index in Oracle.

```
CREATE INDEX SORTSM_BIDES_LNAME
 ON DBSORT.SORTSM_BIDES (LNAME)
           TABLESPACE DBSORT_INDEX
 STORAGE (INITIAL 250K NEXT 100K PCTINCREASE 0);
eof
;;
```

Oracle Grants

Grants assign privileges on objects to database roles. In Oracle the following object-level privileges are available:

- **Select:** The object may be queried
- **Insert:** Records may be added to the object
- **Update:** Existing records in the object may be changed
- **Delete:** Existing records in the object may be deleted
- **Reference:** A table may be used (referenced) to validate a foreign key or domain key constraint

Two other object-level privileges exist in Oracle. They shall not be granted to roles or to users. These privileges are:

- **Index:** An index may be created on the object
- **Alter:** The object may be altered. Columns can be added or deleted and their definitions can be modified. Constraints can be created, modified, or deleted. This grant implicitly grants the right to delete the object as well.

The following script excerpt performs grants in Oracle.

```
sqlplus -silent DBSORT/${DBO_PWD} <<eof
GRANT DELETE,INSERT, SELECT,UPDATE ON DBSORT.
SORTSM_BIDES to DBSORT_RW;
GRANT SELECT ON DBSORT. SORTSM_BIDES to DBSORT_RO;
eof
;;
```

Oracle Database Roles

An Oracle role is a set of privileges that can be granted to users or to other roles. An Oracle user can take on multiple roles. An Oracle user can have multiple active roles or can have some roles active while others are inactive. The DBA creates roles, not the owner. Either the owner or the DBA can assign grants, as discussed above, to the role. The following script excerpt provides an example of the creation of two Oracle roles.

```
CREATE ROLE DBSORT_RO IDENTIFIED BY DBSORT_RO;
CREATE ROLE DBSORT_RW IDENTIFIED BY DBSORT_RW;
```

Oracle Environment Variables

Key environment variables the Oracle system uses are defined during the installation of the Oracle COTS RDBMS Server segment. These environment variables set a common environment for all users, define the Oracle system home directory, and uniquely identify the Oracle database instance.

The following environment variables are set during the installation of the Oracle COTS RDBMS Server segments.

```
ORACLE_HOME         /h/COTS/ORACLE
ORACLE_SERVER       <name of data server host>
ORACLE_SID          GCCS
TWO_TASK            T:${ORACLE_SERVER}:${ORACLE_SID}
path                ($path:${ORACLE_HOME}/bin)
```

These environment variables are accessed through two initialization files on the application servers or client platforms. The files are named `coraenv` for C Shell and `oraenv` for Korn or Bourne Shells.

Oracle Synonyms

The Oracle RDBMS provides synonyms as alternate names for tables, views or other database objects. Private synonyms are user-owned aliases. Public synonyms are system wide aliases.

Developers shall not use Oracle Public Synonyms to identify data tables or views. Public Synonyms belonging to one segment may now, or in the future, conflict with object names belonging to other applications or to the DBMS. Applications should reference objects using full object specifications (owner.table.column).

COE Tool Considerations

Some of the COE tools require special considerations when Oracle is used. This subsection lists the special considerations by tool.

COECreateDS

For an Oracle implementation, `COECreateDS` creates the tablespaces and the database owner account. The database owner account created by this service has an initial password equal to the account name. The account has connect privileges, and has unlimited quotas on the tablespaces created by `COECreateDS`. Its default tablespace is set to the first '`DATA`' tablespace created by the service. If no '`DATA`' tablespace is created by the service, then the first named tablespace is set as the default.

COEExtendDS

For an Oracle implementation, `COEExtendDS` adds additional storage to a tablespace. If the new storage requirements can be serviced then the tablespace is altered by allocating the disk storage to the tablespace.

Sybase RDBMS

This section contains information specific to the Sybase Database Server Segment and to application database segments written for the Sybase RDBMS. It also provides information for developers of application segments that will access Sybase databases.

'`sa`' is the Sybase DBO for all databases. Each segment creates its own database and then grants 'create object' privileges to the segment's DBO account.

Log archiving is accomplished by the Sybase Backup Server.

Sybase Server Segment

The Sybase COTS RDBMS Server segment contains the software required to install and maintain an instance and a database, and the software to support both client/server and server/server communications across the network. Installation of this segment also creates the database files, a master database, log files, temporary storage, and internal data structures that make up a core database to be expanded and populated by other database segments.

The core database is installed with an `sa` account that is the owner of both the core database and the database administrator. Security measures mandate that the password of the `sa` account be changed upon successful installation of this segment. The use of scripts or applications containing hard coded passwords for these accounts, or any other Sybase database accounts, is prohibited.

The Sybase DBMS services segment contains the executables for the RDBMS, network services and the core database instance. The default owner and database administrator is the '`sa`' account. The password of the `sa` account must be changed after the server segment is installed.

The Sybase server segment also sets up the initial database configuration. The initial database includes the master and model databases, and the backup server.

The Sybase 'master' and DBO databases are on mirrored disks. The Backup Server is on a separate set of mirrored disks and is also configurable for off-line archiving to 'dump device.'

Sybase provides separate Logs in each database for online recovery. Logs are archived using the Backup Server to support off-line recovery. Developers are responsible for creating the Logs in their DBO database during the segment's `PostInstall` using COE`CreateDS` or the appropriated database definition script.

Sybase Client Segment

The Sybase COTS RDBMS Client segment contains the Sybase RDBMS executables for accessing the database, the tools and utilities that support data storage, retrieval and backups, and software to support client/server communications across the network. The Sybase client services segment is loaded on each application server. It contains the Sybase client executables. It also has the interfaces file.

Sybase Application Database Segments

There are some special considerations for mission applications that use Sybase services.

DBS_files Directory

For Sybase, segments must create a database and one or more "Sybase segments[12]" within that database for storing their objects. The database segment's prefix will be used as the Sybase database name and as the name of the DBO account. A log device must also be created for the database. It is a violation of the *I&RTS* for an application database segment to store its objects in another application's database segment or in the storage allocated for the database server segment. A segment may create a database that is composed of multiple database devices if the storage requirements are large enough to span physical devices.

Sybase Database Rules

Sybase provides a functionality known as *rules*. Rules are named objects that specify the domain of acceptable values for a particular column or for any column of a custom data type. After a rule is created it can be bound to columns or custom data types.

Database rules should be created before the database tables. The binding of rules can be done when the database tables are created or when constraints are defined. The following script excerpt creates a database rule for UICs with the sample DBSORT data store segment in Sybase.

```
CREATE_RULE()
isql -UDBSORT -PDBSORT -S${DSQUERY} <<eof
create rule UIC_RULE as @UIC like '[NFWME]%'
go
eof
;;
```

Sybase Tables

Sybase tables are implicitly created within the database (schema and storage area) belonging to the database segment. If that database is split up into "Sybase partitions," the create script must explicitly specify the "Sybase partition" that is to store the table. The following script excerpt creates database tables in Sybase.

[12] Sybase uses the term *segment* to define a name to an allocated area of a database device. In this appendix, all references to a physical Sybase segment will be in quotes to differentiate it from a database segment.

```
CREATE TABLE SORTSM_BIDES (
     UIC      VARCHAR (6)    NOT NULL
    ,SECUR    VARCHAR (2)    NOT NULL
    ,TIME     DATETIME  NOT NULL
    ,SERV     VARCHAR (1)    NULL
    ,ANAME    VARCHAR (30)   NULL
    ,UTC      VARCHAR (5)    NULL
    ,ULC      VARCHAR (3)    NULL
    ,SCLAS    VARCHAR (2)    NOT NULL
    ,MJCOM    VARCHAR (6)    NULL
    ,MONOR    VARCHAR (6)    NULL
    ,MAJOR    VARCHAR (1)    NULL
    ,REVAL    VARCHAR (1)    NULL
    ,UDC      VARCHAR (1)    NULL
    ,LNAME    VARCHAR (55)   NULL
    ,COAFF    VARCHAR (2)    NULL
    ,BUPDATE  DATETIME  NULL
    ,TPSN     VARCHAR (7)    NULL)
ON DBSORT_DATA
go
eof
;;
```

Sybase Views

Views should be created after all tables have been created to guarantee that the referenced table objects exist. The following script excerpt creates database views in Sybase to support a change in a column name from SECURITY to SECUR.

```
CREATE VIEW SORTSM_BIDES_1_1
     (UIC, SECURITY, TIME, SERV, ANAME, UTC , ULC,
SCLAS , MJCOM ,
      MONOR, MAJOR, REVAL , UDC, LNAME , COAFF,
BUPDATE, TPSN)
 AS SELECT UIC, SECUR, TIME, SERV, ANAME, UTC , ULC,
SCLAS , MJCOM,
      MONOR, MAJOR, REVAL , UDC, LNAME , COAFF,
BUPDATE, TPSN
 FROM SORTSM_BIDES
go
eof
;;
```

Sybase Constraints

Database constraints are not created when the table is created, but later in the install process. This allows the table creation to occur in any order. When the install process includes a base load of data it can be done faster if the primary keys and indexes are created after the data is loaded. Constraints are also used to bind a rule to columns or custom data types in Sybase tables. The rule must be defined before it can be bound.

The following script excerpts create primary key constraints in Sybase.

```
isql -UDBSORT -${DBO_PWD} <<eof
ALTER TABLE SORTSM_BIDES ADD PRIMARY KEY (UIC)
go
sp_primarykey SORTSM_BIDES, UIC
go
sp_bind rule UIC_RULE, “SORTSM_BIDES.UIC”
eof
;;
```

Sybase Indexes

Indexes must be created after the database tables have been created. A Sybase index should specify a “Sybase segment name.” Clustered indexes should be used unless the table contains dynamic data. Common keys can also be used in Sybase to relate non-primary keys across tables.

The following script excerpts create database indexes in Sybase.

```
CREATE INDEX SORTSM_BIDES_LNAME
     ON SORTSM_BIDES (LNAME)
     ON DBSORT_INDEX
go
eof
;
```

Sybase Grants

Grants assign privileges on objects to database roles. In Sybase the following object-level privileges are available:

- **Select:** The object may be queried
- **Insert:** Records may be added to the object
- **Update:** Existing records in the object may be changed
- **Delete:** Existing records in the object may be deleted
- **Reference:** A table may be used (referenced) to validate a foreign key or domain key constraint

The following script excerpt performs grants in Sybase.

```
GRANT DELETE, INSERT,SELECT, UPDATE ON SORTSM_BIDES to
DBSORT_RW
go
GRANT SELECT ON SORTSM_BIDES to DBSORT_RO
go
eof
;;
```

Sybase Database Groups

Sybase groups are used to manage user privileges and database access. A group is a collection of privileges. Users within a database can only belong to one group at a time. The group definitions must be mutually exclusive. The use of ‘guest’ or ‘visitor’ groups is prohibited. Developers may not grant privileges to ‘public.’

The following script excerpt provides examples for the creation of Sybase groups.

```
isql -UDBSORT -PDBSORT -S${DSQUERY} <<eof
sp_addgroup DBSORT_RO
go
sp_addgroup DBSORT_RW
go
eof
;;
```

Sybase Environment Variables

Key environment variables used by the Sybase system are defined during the installation of the Sybase COTS RDBMS Server segment. These environment variables specify the root of the Sybase installation tree and identify the SQL Server network connection. The environment variable definitions set by the Sybase COTS RDBMS Server segment are as follows:

```
SYBASE          /h/COTS/SYBASE
path            ($path:${SYBASE}/bin)
```

COE Tool Considerations

Some of the COE tools require special considerations when Sybase is used. This subsection lists the special considerations by tool.

COECreateDS

For a Sybase implementation, `COECreateDS` creates the database, the “Sybase segments," and the database owner account. The database name is set to that of ‘`dbo_name`.’ The database owner account created by this service has an initial password equal to the account name. The DBO account is configured to default to the created database.

COEStartDBServer

For a Sybase database server `COEStartDBServer` will check to ensure that all disks that have database files are online before starting up the server.

COEExtendDS

For a Sybase implementation, `COEExtendDS` creates the necessary data segment and adds additional storage to a Sybase database. If the new storage requirements can be serviced then the database is altered by allocating the disk storage to the database.

Informix RDBMS

This section contains information specific to the Informix Database Server and Client segments and to application database segments written for the Informix RDBMS. It also provides information for developers of application segments that will access Informix Databases.

Informix Server Segment

The Informix COTS RDBMS Server segment contains the software required to install and maintain an instance and a database, and the software to support both client/server and server/server communications across the network. Installation of the Informix COTS RDBMS Server segment defines a common environment that provides a central means of updating all user accounts with regard to database changes. Installation of this segment also creates the database files, tablespaces, log files, temporary storage and internal data structures that make up a core database to be expanded and populated by other segments.

The core database is installed with an `informix` account that is the owner of both the core database and the database administrator. Security measures mandate that the password of the `informix` account be changed upon successful installation of this segment. The use of scripts or applications containing hardcoded passwords for these accounts, or any other Informix database accounts, is prohibited.

The Informix COTS RDBMS server segment also sets up the disk storage environment for the creation of database files. The Informix COTS RDBMS server segment installs the initial data segment in the `/opt/informix/DBS_files` directory. Other database segments may place requisite disk storage on other disk drives. The DBA services will provide specific tools to add disk storage for the database.

There are two segments associated with the Informix COTS RDBMS server segment.

1. A development segment that includes Informix OnLine Dynamic Server version 7.1x Development.

2. A runtime segment that includes Informix OnLine Dynamic Server version 7.1x Runtime.

Informix Client Segment

The Informix COTS RDBMS Client segment contains the Informix RDBMS executables for accessing the database, the tools and utilities that support data storage and retrieval and software to support client/server communications across the network.

There are two segments associated with the Informix COTS RDBMS client segment.

1. A development segment that includes Informix ESQL/C version 7.1x Development and Informix -SQL version 6.x Development.

2. A runtime segment that includes Informix ESQL/C version 7.1x Runtime and Informix -SQL version 6.x Runtime.

Informix Application Database Segments

There are some special considerations for mission-application segments that use Informix services.

DBS_files Directory

Data store segments are required to create their own dbspaces for an Informix implementation, their own tablespaces for an ORACLE implementation or their own “Sybase segments” for a Sybase implementation.

The functional storage allocation of a Data Store is implemented in Informix from within the DBMS as one or more ‘dbspaces’ that storing objects. Each segment must create their own database. Segments are prohibited from storing tables and indexes in another segment’s dbspace or the system (root) dbspace.

Informix Tables

Scripts to create Informix tables must specify their size. The tables must be created in dbspaces created by the segment. When creating a table, the Informix ‘in’ parameter must be specified to direct storage allocation. The Informix LOCK MODE parameter must be specified to specify row-level locking. The following script excerpt creates a database table in Informix.

```
CREATE_TABLE)

dbaccess DBSORT <<eof

     create table SORTSM_BIDES (
     UIC       VARCHAR(6)     NOT NULL
     ,SECUR    VARCHAR(2)     NOT NULL
     ,TIME     DATE           NOT NULL
     ,SERV     VARCHAR(1)     NULL
     ,ANAME    VARCHAR(30)    NULL
     ,UTC      VARCHAR(5)     NULL
     ,ULC      VARCHAR(3)     NULL
     ,SCLAS    VARCHAR(2)     NULL
     ,MJCOM    VARCHAR(6)     NULL
     ,MONOR    VARCHAR(6)     NULL|
     ,MAJOR    VARCHAR(1)     NULL
     ,REVAL    VARCHAR(1)     NULL
     ,UDC      VARCHAR(1)     NULL
     ,LNAME    VARCHAR(55)    NULL
     ,COAFF    VARCHAR(2)     NULL
```

```
          ,BUPDATE  DATE           NULL
          ,TPSN     VARCHAR(7)     NULL)
          IN DBSPACE2 LOCKMODE ROW EXTENT SIZE 2000 NEXT
          1000
          eof
     ;;
```

Informix Views

Views should be created after all tables have been created to guarantee that the referenced table objects exist. A segment may create views to support legacy structures as well as application-specific database requirements. The following example script creates a database view in Informix to support a change in a column name from SECURITY to SECUR.

```
     CREATE_VIEW)
          dbaccess DBSORT<<eof
          CREATE VIEW SORTSM_BIDES_1_1
               (UIC, SECURITY, TIME, SERV, ANAME, UTC, ULC,
               SCLAS, MJCOM, MONOR, MAJOR, REVAL, UDC, NAME,
               OAFF, BUPDATE, TPSN)
          AS SELECT UIC, SECUR, TIME, SERV, ANAME, UTC, ULC,
               SCLAS, MJCOM, MONOR, MAJOR, REVAL, UDC, NAME,
               COAFF, BUPDATE, TPSN
          FROM SORTSM_BIDES;
          eof
          ;;
```

Informix Constraints

Database constraints are not generally created when the table is created, but later in the install process. This allows table creation to occur in any order. It also speeds data loading if the primary keys, indexes, and other constraints are created after the data is loaded.

The following script excerpt creates a primary key constraint in Informix.

```
     CREATE_CONSTRAINTS)

     dbaccess DBSORT<<eof
     ALTER TABLE SORTSM_BIDES ADD CONSTRAINT PRIMARY
     KEY(UIC)
     eof
     ;;
```

Similarly, database triggers are installed after all the database procedures are installed. A trigger's body may be coded completely in Informix-SQL, or it could invoke stored procedures to perform the same functions. The use of stored procedures is recommended for performance and maintainability.

Informix Indexes

An Informix index should specify a dbspace. The following script excerpt creates an Informix index.

```
CREATE_INDEX)
     dbaccess DBSORT<<eof
CREATE INDEX SORTSM_BIDES_LNAME
ON SORTSM_BIDES(LNAME)
IN dbspace4
eof
;;
```

Informix Grants

Grants assign privileges on objects to database roles.

In Informix, data access to PUBLIC is prohibited to prevent inadvertent access provided by the default PUBLIC role.

The following script excerpt performs grants in Informix.

```
ASSIGN_GRANTS)

dbaccess DBSORTS<<eof

GRANT DELETE, INSERT, SELECT, UPDATE ON
DBSORT.SORTSM_BIDES TO DBSORT_RW;
GRANT SELECT ON DBSORT.SORTSM_BIDES TO DBSORT_RO;
eof
;;
```

Informix Roles

In Informix, like Oracle, a role is a set of privileges that can be granted to users or other roles. An Informix user can take on many roles.

```
CREATE_ROLE)

dbaccess DBSORT<<eof
create role DBSORT_RO;
create role DBSORT_RW;
eof
;;
```

Informix Environment Variables

Key environmental variables used by the Informix system are defined during the installation of the Informix COTS RDBMS Server segment. These environment variables set a common environment for all users, define the Informix system home directory, and uniquely identify the Informix database. The following variables are defined during the installation of the Informix Server segments.

```
INFORMIXDIR         /h/COTS/INFORMIX
INFORMIXSERVER
ONCONFIG
path                ($PATH:${INFORMIXDIR}/bin)
```

Database Synonyms

A database synonym is functionality provided by both Informix and ORACLE. In Informix, a synonym is an alternate name for a table or a view. Use of synonyms should be avoided. Applications should reference objects using full object specifications (owner.table.column).

Reserved Words

Following is the current, combined list of reserved words from Sybase, Informix, and Oracle which cannot be used (irrespective of case) in naming database elements, etc.

abort
absolute
access
action
add
after
alias
all
allocate
alter
and
any
append
apt
are
arith_overflow
as
asc
assertion
async
at
audit
authorization
avg
backtab
before
begin
bell
between
binary
bit
bit_length
boolean
both
breadth
break
browse
bulk
by
call
callextern
callform
callreport
cancelform
cascade
cascaded
case
cast
catalog
channel
char
character
character_length
charindex
char_convert
char_length
check
checkpoint
close
closesql
cluster
clustered
coalesce
collate
collation
column
comment
commit
completion
compress
compute
confirm
connect
connection
constraint
constraints
continue
controlrow
convert
corresponding
count
create
cross
curindex
current
current_date
current_time
current_timesta
mp
current_user
cursor
cursqlindex
cycle
data
database
datalength
data_pgs
date
datename
datepart
datetime
day
dbcc
deallocate
dec
decimal
declare
default
deferrable
deferred
define
delete
depth
desc
describe
descriptor
diagnostics
dictionary
disconnect
disk
distinct
domain
double
drop
dummy
dump
each
else
elseif
end
end-exec
endtran
enter
entry
equals
errlvl
errorexit
escape
except
exception
exclusive
exec
execute
exists
exit
exitform
exp
external
extract
false
fetch
fetchsql
field
file
fillfactor
first
float
for
foreach
foreign
form
found
from
full
general
get
global
go

goto
grant
group
having
hidden
holdlock
hour
identified
identity
identity_insert
if
ignore
image
immediate
in
increment
index
indicator
initial
initially
inner
input
insensitive
insert
int
integer
interruptsql
intersect
interval
into
is
isolation
join
key
kill
language
last
leading
leave
left
less
level
like
limit
lineno
list
load
local
lock
log
long
loop
lower
match
max
maxextents
mchoice
menu
menubar
min
minus
minute
mirror
mirrorexit
mode
modify
module
money
month
names
national
natural
nchar
new
next
nextquery
no
noaudit
nocompress
noholdlock
nomsg
nonclustered
none
not
nowait
null
nullif
number
numeric
numeric_truncati
on
object
octect_length
of
off
offline
offsets
oid
old
on
once
online
only
open
opensql
operation
operators
option
or
order
others
outer
output
over
overlaps
pad
parameters
parentname
partial
pctfree
pendant
perform
perm
permanent
plan
position
positionform
post
precision
preorder
prepare
preserve
primary
print
printform
prior
private
privileges
proc
procedure
processexit
protected
public
raiserror
raw
rchoice
read
readtext
real
reconfigure
recursive
ref
references
referencing
relative
remote
rename
replace
reserved_pgs
reset
resignal
resource
restrict
return
returns
revoke
right
role
rollback
routine
row
rowcnt
rowcount
rowid
rowlabel
rownum
rows
rule
save
savepoint
schema
schoice
scroll
search
second
Section
select
sensitive

sequence
session
session_user
set
setuser
share
shared
shutdown
signal
similar
size
smallint
some
space
sql
sqlbegin
sqlcode
sqlend
sqlerror
sqlexception
sqlexpr
sqlrow
sqlstate
start
statistics
stripe
structure
submit
substring
successful
sum
switch
switchend
syb_identity
syb_restree
synonym
sysdate
system
system_user
tab
table
temp
temporary
test
text
textport
textsize
then
there
time
timestamp
timezone_hour
timezone_minute
tinyint
to
trace
trailing
tran
transaction
transfer
translate
translation
trigger
trim
true
truncate
tsequal
type
uid
under
union
unique
unknown
update
upper
usage
used_pgs
useform
user
user_option
using
validate
value
values
varchar
varchar2
variable
varying
view
virtual
visible
wait
waitfor
when
whenever
where
while
with
without
work
write
writetext
year
zone

Data Type Compatibility

Developers should use DBMS-provided data types that conform to ANSI/ISO SQL standards and are common across the different COTS DBMS This ensures consistency and portability of database objects and their elements.

Developers shall not use machine-dependent data types (e.g., float) as their behavior across different host computers cannot be predicted. Table F-3 provides a cross reference of Informix, Oracle, and Sybase data types.

Sybase	Oracle	Informix	Range
Integers:			
smallint	smallint	smallint	32,767 to -32,768
int	int	int	2^{31}-1 to -2^{31}
Decimals:			
numeric(p,s)	number(p,s)	numeric(p,s)	10^{38}-1 to -10^{38}
decimal(p,s)	number(p,s)	decimal(p,s)	10^{38}-1 to -10^{38}
float(p) *	float(b)	float(p)	machine-dependent
double precision	double precision	double precision	machine-dependent
real	real	real	machine-dependent
Date/time:			
datetime	date	datetime	valid to seconds only
Character:			
char(n)	char(n)	char(n)	255 chars or less, fixed length
varchar(n)	varchar(n)/varchar2(n)	varchar(n)	255 chars or less, variable length
nchar	nchar	nchar(n)	255 chars or less, fixed length
nvarchar	nvarchar	nvarchar(n)	255 chars or less, variable length
text(n)	long(n)	text	2^{31}-1 chars or less
Binary:			
image	raw	Byte	2^{31}-1 bytes

Table F-3: DBMS Data Types

This page is intentionally blank.

Appendix G: Vendor-Specific DCE Considerations

This appendix contains vendor-specific and version-specific DCE information. It will be updated as either vendors modify DCE products, or as the definition of DCE changes.

DCE 1.1 Features

DCE Version 1.1, released in November 1994 by the Open Software Foundation (OSF), is primarily an "improvement" release that makes DCE easier to deploy and use. There are improvements in serviceability and administration, and Version 1.1 provides more nearly complete compliance with international standards. One administrative improvement is the DCE Control Program (`dcecp`) with a task-oriented command language that combines multiple, complex administrative steps into a single command. Version 1.1 also includes a DCE Host Configuration Agent to automate remote start-up/shut-down and provide remote access to security and configuration data.

Performance improvements in Version 1.1 include the following:

- support for consistent auditing across both the time and security services,
- a new IDL compiler features to increase performance and support internationalization,
- Generic Security Services API (GSSAPI) support for non-RPC applications,
- improvements to the Global Directory Services to improve programming and administration, and
- security registration extensions to support non-UNIX systems.

Note: Many DCE texts do not yet address DCE Version 1.1. Developers should consult OSF or vendor literature to make full use of the DCE Version 1.1 features available in the DII COE.

DCE 1.1 features of particular interest are described below. These features may be of benefit to DII application developers. For more information refer to the *OSF/DCE Application Development Guide - Introduction and Style Guide*.

Serviceability

Serviceability is a set of features that provides a uniform means of managing a server's diagnostic information. DCE defines a set of levels and a means of controlling the destination of generated output based on these levels. By incorporating these interfaces, an application can be better supported/serviced and diagnosed when necessary.

- Production-quality DII servers shall make use of the serviceability features to simplify the presentation of management information.

Note: Serviceability initialization is included within the initialization interfaces provided in the DII COE**.**

DCE Messaging

DCE incorporates a facility for internationalizing textual messages. Messaging is almost always used for client applications, since the messaging API's deal with the internationalization of messages intended for display to the user. If a DII application is intending to use a non-US/ASCII language, this facility should be considered.

Backing Store/Database Library

DCE provides a library of routines for managing a persistent backing store or database of objects. Any data structure that can be described in IDL can be stored in the database, indexed by key or by UUID. The primary purpose for the routines is for servers to store information that must persist between executions. The fact that the information is encoded in IDL ensures that it is stored in a format that is transportable between heterogeneous systems. If a requirement for this functionality exists, it should be used. These functions are not efficient or robust enough to be used for general database programming.

Ada

Several organizations have developed Ada bindings to DCE services. GTE has created a set of Ada bindings to the DCE RPC and threads as part of the U.S. Army Common Hardware and Software-2 (CHS-2) program. These are “thin” bindings; they are one-for-one implementations of the functions in the RPC and Threads libraries, with appropriate conversions for composite types (e.g., between Ada `Strings` and C `char* s`). The host/target was a Sun SPARCstation 20 (also tested on a SPARCstation 10) using SunAda 2.1(g) and Transarc DCE 3.0 (b) libraries.

One of the biggest problems working with Ada and DCE is the issue of threads. Some organizations have been working with Ada vendors to deal with threads and Ada bindings. The U.S. Army, in the CHS-2 program, has done work on the Ada/DCE issues.

Contact the DII COE Chief Engineer for more information on Ada and DCE.

COTS Development Tools

There are a number of COTS products that can be used in the development of DCE applications. The list presented here is not intended to be exhaustive and only provides a brief overview of the products themselves. Current detailed information is available from the OSF World-Wide-Web page at *http://www.osf.org*, or from specific vendors.

> **Note:** This section is provided for information only. The mention of a product does not constitute an endorsement of the product, and the omission of a product does not imply a lack of suitability of the product.

Table G-1 lists several commercially available products to aid in maximizing the use of DCE, and simplifying development. Products are presented alphabetically by product name.

PRODUCT NAME	COMPANY	FUNCTION
CodeCenter & ObjectCenter	Centerline Software, Inc.	Programming Environment
Distributed **A**pplication **T**est **E**nvironment (DATE)	BULL Worldwide Information Systems	Application Test Tool
Dcegen	Learning Works, Inc.	Code Generating Tool
Encina	Transarc Corporation	Development Tools
Entera	Open Environment Corporation	Integration & Development Environment
HP OODCE/9000	Hewlett Packard Corporation	Development Tools
Insure++	ParaSoft Corp	Testing Tool
Micro Focus COBOL	Micro Focus	COBOL Toolkit for DCE
OBJECTIQ-DF	Hitachi America, Ltd	Development Environment
PowerBuilder	Powersoft Corporation	Development Environment
Purify, PureCoverage, Quantify, Purelink	Pure Software	Development Tools
RPCpainter for PowerBuilder	Greenbrier & Russell	Development Tool

Table G-1: DCE-Related COTS Products

CodeCenter and Object Center

(Centerline Software, Inc.)

CodeCenter is used for prototyping, building, testing, debugging, enhancing, and maintaining UNIX C programs. It includes CenterLine-C, an incremental linker, an automatic runtime and static error detection system, an integrated debugger, dynamic graphical browsers and other tools in an integrated system. ObjectCenter is an advanced C++ compilation system. It includes a C++ runtime error detection system, a C++ and C interpreter, an incremental linker, and the CenterLine C++ compiler. DCE applications can be developed using the CenterLine-C compiler and the ObjectCenter C++ compiler. Both compilers have been tested with Transarc's DCE product offerings for the SUN platform.

Platforms supported:	HP HP-UX SUN SunOS SUN Solaris
For more information contact:	Coco Jaenicke Centerline Software, Inc. 10 Fawcett Street Cambridge, MA 02138

Phone: (617) 498-3377
E-mail: *coco@centerline.com*

Distributed Application Test Environment (DATE)
(Bull Worldwide Information Systems)

DATE is a test environment that generates test drivers. Developers of DCE applications define the client/server interface using the Interface Definition Language (IDL). DATE processes the IDL file to automatically generate a client test Driver. The test drivers provide:

- interactive test capabilities through a graphical user interface (GUI),
- playback and record facilities,
- performance measurements, and
- automated regression testing.

Platforms supported: IBM AIX
(Check with the vendor for other platform information.)

For more information contact:	Diane Riemer Bull Worldwide Info. Systems 300 Concord Road Billerica, MA 01821

Phone: (508) 294-4366

Dcegen
(Learning Works, Inc.)

Dcegen is a code generation tool that creates DCE client and server code. The dcegen tool allows the user to create the DCE parts of the client and server program by selecting the type of security and the registration methods to be employed in the application. The dcegen program reads parameters selected in a configuration file and creates the server registration and access checking code which the user links with the server half of the application. It creates the client half of the DCE code by automatically assigning the security methods chosen to the binding handles requested. This tool performs the same functions as the rpcgen tool in the ONC or Sun RPC system.

Platforms supported:	HP HP-UX IBM OS/2 & AIX MS Windows 3.x SUN SunOS & SUN Solaris
For more information contact:	Clay Boyd Learning Works, Inc. 11403 Taterwood Drive Austin, TX 78750-2538
Phone: (512) 258-2729	**Fax:** (512) 258-1215

Encina
(Transarc Corporation)

The Encina product family builds upon the basic services supplied by the OSF DCE, and extends DCE to provide a rich set of facilities for distributed transaction processing. At a high level, these facilities are divided into the Encina Toolkit, which implements the fundamental services for executing distributed transactions and managing recoverable data, and various Encina extended services.

The components that make up the Encina Toolkit are the following:

- **Encina Base Services**. The Encina Base Services module provides services that permit a node to initiate, participate in, and commit distributed transactions.

- **Encina Server Core**. The Encina Server Core provides facilities for managing recoverable data (i.e., data that is accessed and updated transactionally).

The key components of the Encina Extended Services are:

- **Encina Monitor**. The Encina Monitor is a full-featured transaction processing monitor that provides a powerful, reliable environment for the development, execution, and administration of distributed transaction processing applications.

- **Encina Structured File Server (SFS)**. The SFS is a record-oriented file system that provides full transactional integrity, high performance, and log-based recovery for fast restarts.

- **Encina Recoverable Queuing Service (RQS).** RQS enables the transactional enqueuing and dequeuing of data, allowing transactional tasks to be queued for later processing while ensuring that system failures do not result in lost information. RQS provides multiple levels of priority, and readily scales to support large numbers of users and high volumes of data.

· **Encina PPC Executive.** The PPC Executive supports transactional peer-to-peer communications using the CPI-C and CPI-RR application programming interfaces to provide LU6.2 connectivity over TCP/IP.

It has been recommended that Encina be implemented with DCE to provide basic services like queuing and transaction monitoring. Developers who anticipate the need for Encina services should contact the Engineering office to determine Encina availability.

For more information contact:	John Ryan Federal Account Executive Transarc Corporation 6551 Loisdale Court, Suite 950 Springfield, VA 22150
Phone: (703) 924-0283	**Fax:** (703) 924-9586
E-mail: *jryan@transarc*	**URL:** www.transarc.com

Entera
(Open Environment Corporation)

Entera is a heterogeneous integration environment for developing DCE and Encina server applications. Entera uses a three-tier client/server architecture based on the OSF DCE to enable the development and deployment of heterogeneous integrated applications. Entera generates the interfaces that tie applications together. Entera can be used to:

· encapsulate existing applications to convert them into DCE application services,
· generate new DCE application servers, and
· generate DCE interfaces for popular GUI tools and languages.

Entera provides DCE stub support for languages such as Visual Basic, Powerbuilder, Smalltalk and COBOL.

Platforms supported:	DEC UNIX & DEC Windows NT HP HP-UX IBM OS/2 & IBM AIX MS Windows 3.x & NT SUN Solaris
For more information contact:	Anne Thomas Open Environment Corporation 25 Travis Street Boston, MA 02134
Phone: (617) 562-0900	**Fax:** (617) 562-0038
E-mail: *inf@oec.com*	

HP OODCE/9000
(Hewlett Packard)

HP OODCE frees developers from the DCE API and enables them to access and manage DCE services through a set of pre-defined C++ objects that encapsulate multiple DCE commands. A programmer can use OODCE/9000 objects as building blocks for rapid DCE development. Examples of OODCE/9000 objects include the Global Server Object and Access Control List Database Manager Object.

Platforms supported: HP HP-UX
Other platforms supported--check with Vendor.

For more information contact: Bruce Talley
Hewlett Packard
19111 Pruneridge Avenue
Cupertino, CA 95014

Phone: (408) 447-1830 **Fax:** (408) 447-1831
E-mail: *Bruce-Talley@HP4700.DESK.HP.COM*

Insure++
(ParaSoft Corporation)

Insure++ automatically detects large classes of programming and runtime errors. It is able to debug multi-threaded applications in a DCE environment. Insure++ can be used to find the following types of errors:

- memory reference errors,
- library interface errors, and
- algorithmic anomalies, bugs, and deficiencies

Platforms supported: SUN Solaris
IBM AIX
DEC Alpha
(Check with the vendor for other platform information.)

For more information contact: Scott Timmons
Parasoft Corporation
2031 S. Myrtle Ave.
Monrovia, CA 91016

Phone: (818) 305-0041 **Fax:** (818) 305-9048
E-mail: *sct@parasoft.com* **URL:** www.parasoft.com

<u>Micro Focus COBOL</u>

(Micro Focus)

MicroFocus has toolkits to provide support for COBOL programming for DCE. The COBOL toolkit for DCE includes application development tools to enable existing COBOL applications to use DCE without having to modify source code or define interfaces. It also includes a COBOL API to DCE services to ease DCE application development, a high-level API to make DCE programming easier, and a COBOL IDL compiler to allow definitions of interfaces using COBOL data types. The Micro Focus COBOL Toolbox for UNIX includes a cross-platform COBOL compiler, an interactive debugger and an integrated set of programmer productivity aides and utilities.

Platforms supported:	Windows UNIX

(Check with the vendor for more specifics.)

For more information contact:	Micro Focus Sales 2465 East Bayshore Road Palo Alto, CA 94303
Phone: (415) 856-4161	**Fax:** (415) 856-6134

<u>OBJECTIQ-DF</u>

(Hitachi America, Ltd)

OBJECTIQ-DF (distributed facility) is based upon the OSF DCE remote procedure call. Client programs call procedures in server (remote) programs as if the procedures were at the client. All objects appear local to a developer. Messages are passed between objects whether they are resident at the client or server. This is accomplished during development as OBJECTIQ-DF generates IDL (Interface Definition Language) descriptions saving the developer from having to code low-level IDL programs.

Platforms supported:	HP HP-UX IBM-AIX SUN Solaris
For more information contact:	Steven L. Wentworth Hitachi America, Ltd 437 Madison Avenue 33rd Floor New York, NY 10022
Phone: (212) 702-1500	**Fax:** (212) 751-6368

E-mail: *Swentworth@hal-com.mhs.compuserve.com*

PowerBuilder
(Powersoft Corporation)

PowerBuilder is a powerful, easy to use environment for building graphical client/server applications. PowerBuilder is designed for MIS software developers to create applications that integrate fully with high-performance, relational database servers in a transaction processing environment. PowerBuilder applications can invoke DCE RPC-based services using the Windows version of DCE.

Platforms supported:	DEC Windows NT MS Windows 3.x & NT SUN Solaris
For more information contact:	Corporate Sales Powersoft Corporation 561 Virginia Road Concord, MA 01742
Phone: 800-395-3525	**Fax:** (508) 369-3997

Purify, PureCoverage, Quantify, Purelink
(Pure Software)

Pure Software makes a family of products for developing, debugging, and testing C and C++ programs in a DCE environment. The suite includes:

- **Purify**. Purify provides runtime error checking and memory leak detection throughout C and C++ programs. Purify is thread aware and can provide DCE developers runtime error information on their threaded applications.
- **PureCoverage**. PureCoverage identifies code that has not been executed in applications, including third-party and shared libraries. PureCoverage is thread safe and can be used with threaded applications.
- **Quantify**. Quantify is a performance analysis tool that provides accurate and comprehensive performance data for an application. It is thread aware and provides composite performance information of all the threads defined in an application as well as performance data on a specific thread.
- **PureLink.** Purelink is a linker that cuts build time by identifying what has been changed and only relinks those portions of code.

Platforms supported:	HP HP-UX SUN SunOS SUN Solaris
For more information contact:	Pamela Roussos

	Pure Software 1309 South Mary Avenue Sunnyvale, CA 94087
Phone: (408) 524-3033 **E-mail:** *Roussos@pure.com*	**Fax:** (408) 720-9200

<u>RPCpainter for PowerBuilder</u>
(Greenbrier & Russell)

RPCpainter supports 3-tiered architecture by offering a seamless interface between PowerBuilder and the DCE RPC. RPCpainter functions include:

· a graphical tool for preparing and editing remote procedure interface definitions,
· automatic generation of RPC stubs for the server,
· automatic generation of RPC objects in a PowerBuilder Library for the client,
· automatic population of DataWindows with RPC results, and
· automatic upload of DataWindow changes through RPCs

Platforms supported:	HP HP-UX IBM AIX MS Windows 3.x & NT SUN Solaris
For more information contact:	Scott Mitchell Greenbrier & Russell, Inc. 1450 E. American Lane Suite 1640 Schaumburg, IL 60173
Phone: (708) 706-4000	

Glossary

Account Group: *A template for establishing a runtime environment context for individual operators*. Account groups are typically used to do a high-level segregation of operators into system administrators, security administrators, database administrators, or mission-specific operators.

Advanced Interoperability: *A level of interoperability characterized by shared data between applications, including shared data displays, and information exchange through a common data model*. This level provides for sharing of information in a distributed, but localized environment, and for sharing of applications.

Affected Account Group(s): *The account group(s) to which a segment applie*s. Functionality provided by the installed segment will normally appear to the operator as new menu items or icons in the affected account group(s).

Aggregate Segment: *A collection of segments grouped together, installed, deleted, and managed as a single unit*. Aggregates are a convenient way for grouping segments that need to be developed and managed separately, but which must be presented to an operator as a single collection of functions.

Application Programmer Interface (API): *A programmer's guide that describes the data structures, function calls, and services provided by the COE, and how to write software modules that interface with and use COE services*. The definition of what actions an API performs is programming-language neutral and may includes traditional function (C/C++) or procedure (Ada) interfaces as well as command-line interfaces. The exact syntax of an API is language-specific. Data structures are included in the definition as are the definition of objects for services that are written in an object-oriented model.

Approved Software: *Software that has been tested as compatible with the COE*. An approved products list might contain Oracle, Sybase, WordPerfect, Kermit, etc. In this context, approved software implies only that the software has been tested and confirmed to work within the DII COE. It does *not* imply that the software has been approved or authorized by any government agency for any specific system.

Architecture: *The structure of components, their relationships, and the principles and guidelines governing their design and evolution over time* (IEEE STD 610.12). Three types of architectures are defined: operational, technical, and systems.

Assigned Directory: *The unique directory assigned to a segment at segment registration time that is to be the segment's home directory*. A segment is normally completely contained within its assigned directory and may not modify any files outside its assigned directory without going through facilities provided by the COE.

Basic Interoperability: *A primitive level of interoperability characterized by peer-to-peer connected systems that allow basic exchange of homogenous data (e.g., email, formatted messages), and allows for basic collaboration*. This level of interoperability is achievable by relatively simple interfacing techniques, and by use of standard office automation products that provide data import/export functions for handling data from another product.

Broker: *An intermediary that coordinates and manages the requests between clients and servers*. Brokers are generally discussed in the context of CORBA (Common Object Request Broker Architecture) or other object-oriented approaches, but the usage need not be so restrictive. An example of a broker operation is to connect a client requesting some service to an available server located at some arbitrary location on the LAN.

Client: *A computer program, such as a mission application, that requires a service.* Clients are consumers of data while servers are producers of data.

Client/Server: *A particular kind of computing architectural model in which consumers (clients) and producers (servers) cooperate to create an application*. Clients request services from servers, while servers may service one or more clients simultaneously. A typical example of a server is a database server. An example of a client is a software component that allows an operator to prepare a query, pass the query to the database, and then display the results to the operator.

COE-Component Segment: *A segment that is contained within the COE*. All software in COE-based systems is packaged as a segment, including those within the COE itself. Strictly speaking, "COE component" is a segment attribute rather than a separate segment type. Segments are specifically identified as COE components because specialized processing is performed on them during software installation, and they are handled more rigorously in the development cycle.

COE Kernel: *That subset of the COE-component segments that is required on all platforms*. As a minimum, this consists of the operating system, windowing software, security, segment installation software, and an Executive Manager. The DII COE is designed to minimize the size of the kernel so that minimal resources are required at each platform. Definition of the kernel is independent of whether the platform will be used as a database server, an applications server, or a client platform.

Commercial Off-The-Shelf Software (COTS): *Software that is available commercially*. Examples include versions of UNIX, X Windows, or Motif, as well as standard products such as Oracle, Sybase, and Informix.

Common Operating Environment (COE): *The collection of standards, specifications, and guidelines, architecture definition, software infrastructure, reusable components, APIs, methodology, runtime environment definition, reference implementation, and methodology that establishes an environment on which a system can be built*. The COE allows segments created by separate developers to function together as an integrated system. The COE is the vehicle that assures interoperability through a reference implementation that provides identical implementation of common functions. It is important to realize that the COE is both a standard and an actual product (e.g., reference implementation) composed of reusable software components built according to a set of open standards and specifications.

Community Files: *Files that reside outside a segment's assigned directory*. To prevent conflict among segments, community files may be modified only through the COE installation tools. Examples of community files include `/etc/passwd`, `/etc/hosts`, and `/etc/services`.

Compliance: *An integer value, called the compliance level, which measures (a) the degree to which a segment or system achieves conformance with the rules, standards, and specifications described by the COE, (b) the degree to which the segment or system is suitable for integration with the DII COE reference implementation, and (c) the degree to which the segment or system makes use of COE services*. Compliance is measured for segments and COE-based systems. It is important to note that compliance is a matter of degree. It is usually not an “all or nothing” proposition, but it is possible to fail to meet any COE objectives and hence be declared totally non-compliant. Compliance is measured in four areas, called compliance categories. The four categories are Runtime Environment, Architectural Compatibility, Style Guide, and Software Quality. The higher the level of compliance, the higher the level of interoperability with other COE-based systems.

Compliance Category: *One of the four areas (Runtime Environment, Style Guide, Architectural Compatibility, Software Quality) in which DII compliance is measured*. These four categories form a spectrum for measuring compliance with regard to COE compatibility and degree of interoperability (Runtime Environment), user friendliness (Style Guide), product longevity (Architectural Compatibility), and program risk (Software Quality).

Compliance Level: *The degree to which a segment is DII-compliant within a specific compliance category*. Compliance levels are integer values only.

Component Database: *Individual database within a multi-database design*.

Composite Compliance: *The DII compliance value assigned to a collection of segments*. The purpose of a composite compliance value is to describe the degree of compliance a system achieves when it may contain COE-component segments that themselves are not Level 8 compliant. The composite compliance level for an arbitrary collection of segments is the compliance level of the *least* compliant segment. Chapter 2 provides formulas for computing the composite compliance level for a COE-based system, and for systems which contain both COE and non-COE based platforms.

Configuration Control Board (CCB): *The organization responsible for authorizing enhancements, corrections, and revisions to the COE, or to a COE-based system*. CCBs exist for the purpose of controlling changes made to a system. DISA chairs the CCB for the DII COE and for its own systems (e.g., GCCS, GCSS, ECPN). Other services and agencies may use the same approach for controlling changes to their COE-derived systems.

Configuration Definition: *A hierarchical representation of a collection of segments that are to be installed*. Configuration definitions are organized into distributions, folders, configurations, bundles, and segments. The purpose of configuration definitions is to allow pre-definition of the segments that are to be installed at a site broken down into whatever groups are meaningful to the site (e.g., workspace, platform usage).

Data Store: *A functional grouping of data storage based on the type of information, or use of the storage*. A data store in Sybase is a database partition, while in Oracle it is a tablespace.

Database: *A structured set of data, managed by a DBMS, together with the rules for accessing the data*. A database is more than just a collection of data files.

Database Administrator: *The DBMS user account for DBMS administration functions.* Sybase uses "`sa`" for the DBA account while Oracle uses "`system`."

Database Management System (DBMS): *Software to manage concurrent access to shared databases.* There are several techniques for organizing and managing databases. Most commercial products are relational database management systems but new technology advances are making object-oriented database management systems a reality.

Database Owner: *The DBMS user account that is the creator or owner of the data objects that are part of the data store segment.*

Database Schema: *The specific data view or design of a particular database.* One of the powerful features of a database system is that different applications can have their own view of the database by defining different schema. The database management software handles the details of mapping the actual data representation on disk into the view the application requires.

Database Segment: *A standard method for packaging a physical database for incorporation into the COE/SHADE.* Database segments are packaged like any other COE segment.

Database Services User: *A special DBMS user account accessed only by an autonomous application such as a message processor that provides services to other users via the DBMS.*

Database Session: *An individual connection between an application program and a database management system.* Sessions are defined as a security measure for database accesses.

Descriptor Directory: *The subdirectory* `SegDescrip` *associated with each segment.* This subdirectory contains descriptors that provide information required to install the segment.

Descriptors: *Data files (contained in the segment's descriptor directory) that are used to describe a segment to the COE.* The software installation and integration process uses descriptor directories and their descriptor files to ensure DII compliance. Descriptor files permit automated integration and installation and are the technique for segments to "self-describe" themselves to the COE.

Development Environment: *The software environment required to create, compile, and test software.* This includes compilers, editors, linkers, debug software, and developer configuration preferences such as command aliases. The development environment is distinct from the runtime environment, and **must** be separated from the runtime environment, but it is usually an extension of the runtime environment.

Distributed Database: *A database whose data objects exist across, or are fragmented across, multiple computer systems or sites.* The ability to distribute a database across multiple sites is key to collaborative planning in a number of problem domains including GCCS, GCSS, and ECPN.

Distributed Processing: *The ability to perform collaborative processing across multiple computers*. Among its many benefits, this capability allows processing load to be distributed across computers within the network. It also allows applications running on one platform to access capabilities that it may not have the hardware resources to host itself. DCE, CORBA, and DCOM are all techniques for performing distributed processing.

Environment: *In the context of the COE, all software that is running from the time the computer is rebooted to the time the system is ready to respond to operator queries after operator login*. This software includes the operating system, security software, installation software, windowing environment, COE services, etc. The environment is subdivided into a runtime environment and a software development environment.

Environment Extension: *The process of a segment issuing requests to the COE tools to expand the environment*. The principle of environment extension is important because it allows for segments to be easily added or removed, and it eliminates unarbitrated environmental conflicts between segments.

Environment Extension File: *A file that contains runtime environmental extensions for the COE*. Segments use extension files to add their own environment variables to the runtime environment, additions to the X Windows environment, etc. Environment extension files are the technique for encapsulating a segment's contribution to the runtime environment so that its environment effects can be easily added or removed.

Fragmentation Schema: *The distribution design for a distributed database*. The idea is that a schema (e.g., an application's view of the database) may actually require access to data that is distributed across computers or networks.

Government Off-The-Shelf (GOTS) Software: *Software developed through funding by the US Government*. The DII COE contains both COTS and GOTS segments.

Integration: *The process of combining components, usually hardware and software, into a new, larger component to achieve some architectural requirement*. Integration requires resolution of compatibility issues between components that are to be interconnected. Integration attempts to allow sharing of a common resource (such as data) without the need for intermediate translations from one format to another. Note that the COE is a technique for achieving integration that ensures interoperability.

Interfacing: *The process of two components or systems exchanging information by first translating the information into an intermediate, agreed-upon format*. For example, track information is transmitted between FOTC participants by generation of an OTH-GOLD message on the sending system, and parsing the message on the receiving system. If the two systems were truly integrated, the translation to/from OTH-GOLD would not be required. Note that standards are the technique for specifying how interfacing can be accomplished.

Intermediate Interoperability: *A level of interoperability characterized by a client/server environment with standardized interfaces and distributed computing services that allow for exchange of heterogeneous data (e.g., maps with overlays, annotated images), and advanced collaboration*. This level of interoperability is achievable with implementation of "cut and paste" between applications, through World-Wide-Web technology, and through basic use of DII COE features.

Interoperability: *The ability of two or more systems or components to exchange and use information* (IEEE STD 610.2). This definition is extended in the context of a COE to include levels of interoperability, and relate interoperability to interfacing (lowest, least desirable level) versus true integration (highest, most desirable level).

Mission-Application Segment: *A segment that uses the services of the COE to provide functionality that is specific to a mission domain*. Mission-application segments are external to the COE and are built on top of the COE to create a new capability.

Multi-Database: *A collection of autonomous databases*. The databases may exist on a single platform, or be distributed across multiple platforms.

Open System: *A system that implements sufficient open specifications for interfaces, services, and supporting formats to enable properly engineered applications software: (a) to be ported with minimal changes across a wide range of systems, (b) to interoperate with other applications on local and remote systems, (c) to interact with users in a style that facilitates user portability, and (d) to enable users to increase processing power as their functional needs grow, without the need to re-write applications (i.e., scalability)* (POSIX ISO 9945-1). The "openness" of a system is a multi-dimensional measurement that in essence determines how easy it is to change the system from one intended purpose to another.

Operational Architecture: *Descriptions of the tasks, operational elements, and information flows required to accomplish or support a warfighting function. The operational architecture is best thought of as that which identifies the warrior's need* (*JTA*). The operational architecture is closely tied to CONOPS (Concept of Operations) and is often dictated by policy rather than technology. It represents the desired system process and is more of the "what" rather than the "how." An example of an operational architecture is the reporting chain of command from a soldier in the field to the CJTF.

Plug and Play: *The ability for a COE-derived system to be scaled by the addition/removal of hardware or software components*. To have the property of "plug and play," the system must be able to reconfigure itself automatically after component addition/removal with required human intervention limited to no more than a system power down or reboot.

Portability: *The degree to which system components may be transferred from one hardware or software environment to another* (IEEE STD 610.12). High portability means that the transfer occurs with minimal or no modifications. For example, a COE segment which was originally developed on a Solaris platform that can be executed correctly on a Hewlett Packard or Silicon Graphics platform by rebuilding the segment without modifying the source code is highly portable.

Profile: *The subset of the total COE-based functionality that is to be made available to a group of individual operators*. Profiles are defined by the security or system administrator to limit what an operator can access both on a "need to know" basis and as a way to avoid overwhelming the operator with functions that are not useful for his mission area.

Reference Implementation: *The set of software and hardware components that implement a COE*. The reference implementation of a COE may change over time to take advantage of new technologies or to fix problem reports, but the principles governing the COE remain unchanged. A COE is incomplete without a reference implementation because it is the reference implementation that is most responsible for ensuring interoperability. The DII COE is an example of a reference implementation.

Remote Install: *The ability to electronically install segments from a local site (such as the DISA Operational Support Facility) to a remote site (such as USACOM)*. In a "push" mode, the local site initiates and controls the segment installation. In a "pull" mode, the remote site initiates and controls the segment installation.

Reusability: *The ability of a software or hardware component, built to fulfill a requirement for a system built to achieve a particular mission, to be used to meet a similar requirement for another system that is being built to address a different mission need*. Reusability allows system development and maintenance costs to be reduced because they can be shared across multiple programs. The DII COE contains a reuse strategy.

Runtime Environment: *The runtime context determined by the applicable account group, the COE, and the executing segments*. The runtime environment definition is limited to just the context (environment variable settings, X configuration, background processes, task priorities, etc.) required for an application to execute. Environment settings for software development purposes are maintained separately from the runtime environment to avoid runtime conflicts between developer preference settings.

Scalability: *The ability of a system to increase (decrease) functionality , the amount of data which can be processed or stored, and responsiveness without the need to re-write applications through the addition (removal) of hardware and software components*. Scalability is critical in achieving "NCA to foxhole" at reasonable cost.

Script: *A file containing commands to be executed by an operating system shell*. Such files are generally called "scripts" in the UNIX environment, but are typically called "batch files" in the MS-DOS and Windows environment.

Segment: *A collection of one or more software and/or data units most conveniently managed as a unit of functionality*. Segments are defined from the perspective of an operator, not a developer, and are generally defined to keep related units together so that functionality may be easily included or excluded. They are usually defined as functional pieces (e.g., a word processor) that make sense from a system administrator perspective because segments are the lowest level components that can be installed on, or removed from, a platform

Segmentation: *The engineering process of decomposing system components into segments and creating the appropriate segment descriptor files*. Proper segmentation is vital to a good system design and affects how well the component will operate in the resulting system.

Segment Compliance: *The degree to which a segment, whether it is part of the COE or a mission application, is compliant*. Compliance of individual segments is computed in order to calculate overall system compliance.

Segment Descriptor: *An ASCII file which contains information that describes attributes about the segment*. The format of these files is described in the *I&RTS*. Segment descriptor files are processed by various tools in the COE to validate compliance and to perform segment installation.

Segment Descriptor Directory: *The directory,* `SegDescrip`*, which contains the segment descriptor files used to describe a segment to the COE*. Each segment is required to contain a segment descriptor directory in order to be DII-compliant.

Segment Prefix: *A 1-6 alphanumeric character string assigned to each segment for use in naming public symbols*. Segment prefixes are used in any situation where it is important to make sure that there will not be a naming conflict between developers (environment variable names, executable file names, shared library names, etc.). Segment prefixes are not necessarily unique among all segments, but the combination of a segment prefix and segment name is *always* unique among all segments.

Segment Servers: *One or more designated platforms on a LAN that have segments stored on them in a format that can be used for installation on other platforms*. Designation of segment servers greatly simplifies distribution and installation of a system. It also speeds up installation because it can be done from disk across the LAN rather than from slower tape magnetic media.

Server: *A computer program that provides some service*. Servers are producers of data while clients are consumers of data.

Service: *A function that is common to a number of programs, such as performing some extensive calculation or retrieving a category of data*. An example of a service is a function that accepts a request to transform a point from one coordinate system into another.

Session: (See Database Session)

Shared Data Server: *A DII-compliant platform that provides data server functions for multiple mission applications.*

Shared Database Segment: *A database segment that supports the information requirements of multiple applications or across multiple database segments.* Shared database segments are typically mission-or-functionally-oriented, and are generally specific to a limited number of mission domains. Shared database segments are under joint configuration control. An example of such a database segment is a database of logistics drawings for military hardware.

Shared Data Environment (SHADE): *The DII COE strategy and mechanisms for data sharing.* SHADE includes the required data-access architectures, data sharing methodology, reusable software and data components, and guidelines, standards, and specifications for the development and migration of systems that meet the user's requirements for timely, accurate, and reliable data.

Superset: *The total collection of all COE-based segments available to the development community.* The superset includes the COE as well as mission-application segments.

System: *A set of different elements so connected or related as to perform a unique function not performable by the elements alone. The most important and distinguishing characteristic of a system, therefore, is the relationships among the elements.* An example is an automobile. All the individual elements may be in working order but do not individually provide transportation. Transportation only exists when all the elements are connected and function as a whole.

Systems Architecture: *Descriptions, including graphics, of systems and interconnections providing for or supporting warfighting functions* (*JTA*). The system architecture typically relates capabilities to requirements and requirements to standards.

System Compliance: *The degree to which a COE-derived system in total meets COE standards, specifications, and rules.* System compliance is a composite measure of all segments in the system whether they are part of the COE, or are application segments. The measure also includes provisions for components that are not COE-derived (e.g., mainframe components).

Technical Architecture: *A minimal set of rules that govern arrangement, interaction, and interdependence of the system components* (*JTA*). The technical architecture identifies standards and conventions to be used.

Unique Database Segment: *A database segment that is limited in scope and typically used by only one application.* Unique database segments may be shared between applications, but the usage is restricted to a single mission domain. An example of a Unique database segment is a configuration table that an application reads at initialization time. Unique database segments are under the configuration control of the segment sponsor.

Universal Database Segment: *A database segment that represents widespread data requirements and is used by many applications, many database segments, and spans multiple mission domains*. They are typically reference or lookup tables. An example is a database of country-code abbreviations. Universal database segments are under stricter configuration control with DISA and DOD Data Administration coordination.

Universal Interoperability: *An interoperability level characterized by the ability to globally share integrated information in a distributed information space*. Universal interoperability represents the ultimate interoperability goal.

Index